Approved for public release; distribution is unlimited.

MULTIGRID METHODS IN NETWORK OPTIMIZATION: OVERVIEW AND APPRAISAL

by

Javier Nieto,
Lieutenant Commander, Spanish Navy

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH
MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the NAVAL POSTGRADUATE SCHOOL
March 1994

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1994	Master's Thesis

4. TITLE AND SUBTITLE
MULTIGRID METHODS IN NETWORK OPTIMIZATION: OVERVIEW AND APPRAISAL

5. FUNDING NUMBERS

6. AUTHOR(S)
NIETO, Javier

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE
*A

13. ABSTRACT *(maximum 200 words)*

Multigrid methods have been traditionally applied to the solution of certain Partial Differential Equations. However, applications in control theory, optimization, pattern recognition, computational tomography and particle physics are beginning to appear.

This thesis analyzes the application of multigrid methodology to optimization problems. The work is centered on networks. Transportation problems are chosen frequently as reference because they have been the object of some multigrid research. The goal is to establish a basis for development of multigrid-based algorithms.

Optimality conditions in linear programming and networks are reviewed, and a compilation of various multilevel approaches in optimization is presented. Emphasis is on the recent scaling techniques; they add some special insights into solving large network problems efficiently using progressive level of detail. An analysis of the difficulties that these problems present to the multigrid approach reveals that perhaps some abstraction is appropriate when interpreting multigrid components applied to optimization problems (in particular, the concept of grid itself). The idea of implicit ordering is developed and associated with the effectiveness of the multigrid method. These concepts are applied to identify problems that can be solved using multigrid. Finally, suggestions for the development of future multigrid-based algorithms are provided.

14. SUBJECT TERMS .
Multigrid, Multilevel, Decomposition, Scaling, Aggregation, Perturbation.

15. NUMBER OF PAGES * 161

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

ABSTRACT

Multigrid methods have been traditionally applied to the solution of certain Partial Differential Equations. However, applications in control theory, optimization, pattern recognition, computational tomography and particle physics are beginning to appear.

This thesis analyzes the application of multigrid methodology to optimization problems. The work is centered on networks. Transportation problems are chosen frequently as reference because they have been the object of some multigrid research. The goal is to establish a basis for development of multigrid-based algorithms.

Optimality conditions in linear programming and networks are reviewed, and a compilation of various multilevel approaches in optimization is presented. Emphasis is on the recent scaling techniques; they add some special insights into solving large network problems efficiently using progressive level of detail. An analysis of the difficulties that these problems present to the multigrid approach reveals that perhaps some abstraction is appropriate when interpreting multigrid components applied to optimization problems (in particular, the concept of grid itself). The idea of implicit ordering is developed and associated with the effectiveness of the multigrid method. These concepts are applied to identify problems that can be solved using multigrid. Finally, suggestions for the development of future multigrid-based algorithms are provided.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

I wish to express my deepest recognition and admiration to Gordon H. Bradley, Van Endem Henson and Craig W. Rasmussen. They made this thesis the best academic experience I ever have been exposed to.

My special thanks to my wife Ana and my daughters, Maria and Ana. Their support has been invaluable. This work is dedicated to them.

EXECUTIVE SUMMARY

Multi-level processes are found in many military, government and private sector activities. In many cases the multi-level processes are easy to observe because they are inherent in the activity itself. For example, in the shipment of goods from factories to warehouses and then to customers the multi-level nature of the distribution is obvious. In other cases the multi-level processes are in the structure or environment in which the activity is developed. For example, military personnel assignment has multiple levels because of way that jobs are assigned according to specialties or ranks. In still other cases multilevels are introduced where none is inherent in order to break the problem into parts that may be solved more easily than the original. Multilevel problems are virtually all very large and it is usually necessary to develop solution techniques that exploit special structure associated with the multilevel processes in order to effectively construct solutions.

Multigrid methods are a multilevel approach to solving problems. The basic idea is to sample problems at different levels of detail so that their simpler representation on coarser grids can reduce computational effort. They seem to be particularly effective when there is some inherent hierarchical structure or when a hierarchical structure can be induced on the problem. In addition to hierarchical structure there are other problem characteristics that are critical to the success of multigrid methods.

Traditionally, multigrid methods have been applied to the solution of certain Partial Differential Equations, but they are much more widely applicable than just to the numerical solution of differential and integral equations. Applications in such diverse areas as control

The present thesis analyzes the possibilities of applying the multigrid methods to solve optimization problems. The work is centered on network problems, particularly transportation problems. The reason for this choice is that the efficient solution of large network problems is important in many military and industrial activities, and also there has been research on the application of multigrid methods to transportation problems. The thesis does not actually solve any network problem. The goal is to develop a basis for future development of algorithms that take advantage of the powerful characteristics of multigrid to solve large networks and, ultimately, more general optimization problems.

This thesis provides an overview of the basic aspects of optimality in linear programming and networks, and a compilation of various multilevel approaches in optimization. Special emphasis is made on the recent scaling techniques; they add some special insights into the task of solving large network problems in a progressive level of detail and are very efficient computationally. An analysis of the difficulties that those problems present to the multigrid approach reveals that perhaps a high level of abstraction is appropriate when interpreting the application of multigrid components to optimization problems. New ideas to interpret the concept of grid are proposed, and the idea of implicit ordering is associated with the effectiveness of the multigrid method. Finally, these concepts are applied to identify problems to be solved using multigrid. The conclusions provide some new insights and give some suggestions for the development of future multigrid-based algorithms oriented to the solution of optimization problems.

I. INTRODUCTION

A. MOTIVATION

Multilevel algorithms have emerged as the most efficient algorithms for solving certain very large algebraic systems arising from discretizing partial differential boundary value problems. Multilevel approaches have naturally emerged in many branches of computer technology, as in the structured organization of computer hardware, the top-down structured design of software, the pyramidal data structures (trees, heaps, etc.) and many of the most efficient algorithms in computer science, such as fast sorting and others in the "divide and conquer" class of algorithms. The multigrid approach adds some other characteristics to establish a powerful and general framework for the solution of a wide class of problems. This research explores the suitability of such methods to the solution of optimization problems.

B. BASIC CONCEPTS

In the numerical solution of differential equations, each variable is represented by a vector of real values. This means that the solution of the problem can be regarded as an array of vectors, each corresponding to an individual variable. In the simpler one-dimensional case, the solution of the problem is a single vector, which we will refer to as the ***solution vector***.

Considering the one-dimensional case (Hackbusch, 1980), relaxation methods, i.e. Jacobi and Gauss-Seidel iterations, begin with an initial guess of a solution. They then proceed to improve the current approximation by updating steps (iterations). If we could represent the error at each step, it would be a vector of the same number of components as the solution vector. It is referred to as the ***error vector***. At the beginning of the iterative process, the error vector is somewhat arbitrarily constructed. The arbitrariness is related to the choice of a first guess taken as initial solution. Some problems present special behaviors of the error vector along the iteration steps. We are interested in the kind of problems in which the error can be represented by a linear combination of frequency components (i.e. Fourier nodes). Moreover, we are interested in errors that through the iterative process are "smoothed" to end up being formed by only *long waves.* (At this point, it suffices to say that long waves are those with relatively low frequency components). It is hard to detect such structure in the error vector, and even harder to know about its evolution. In fact, to detect this structure, we need to know the error itself, and knowing the error is equivalent to having solved the problem. Again, it is the evolution of the error that attracts our attention to these problems, because it makes them suitable for special solution techniques, known as *multigrid techniques.*

In these problems the error vector, in general, will consist of high frequency oscillatory components, medium frequency components, and low frequency smooth components. Conventional relaxation methods, as those mentioned above, reduce the oscillatory components first. When the remaining error's spectrum is formed essentially by smooth components a progressive slowness in the convergence to the solution is observed. At this point, the iterative processes become extremely slow.

The multigrid approach is to treat each component of the error on the environment (grid) where that component appears to be "oscillatory".

The multigrid approach takes advantage of the following facts:

- Smooth components of the error are well represented on coarser grids. This facilitates solving the problem on a "simpler" domain;
- When transferring smooth components into the next domain (coarser grid), moderately-smooth components of the error become oscillatory components of the error in the new domain. This fact speeds up the iterative process since oscillatory components are reduced faster.

The complementary use of these two ideas constitutes the basis of the multigrid method. Both of them contribute to accelerate convergence and, with the help of some intuition, explain the success of that approach.

Two main tools are used in approaching the solution of problems in multigrid (Briggs, 1987):

(a) Use of coarser grids to obtain initial guesses for the problem on finer grids (***nested iteration***);

(b) Use of approximate solution in fine grids to obtain a ***residual***, defined as the difference between the true and approximated right hand side vectors. The idea is to relax until the error is smooth, then transfer the residual equation to coarser grids to relax on the error, and finally correct the approximate solution in the finer grid after interpolating the error. (***coarse grid correction***).

C. INTERPRETATION OF THE IDEA IN OPTIMIZATION PROBLEMS

The above ideas suggest that a parallel approach could be implemented in solving optimization problems. The basic ideas should be:

(a) *Nested Iteration.-* Form a reduced problem at the simplest possible level. Solve, and use the solution to obtain a good initial guess to the extended problem at the next level. Proceed in this fashion until the finest level is reached.

(b) *Coarse Grid Correction.-* Find an initial guess to the problem, that we will call the "complete problem". Work on this guess to obtain an improvement. Iterate until a point of little advance is reached. Then shift to a "reduced problem", derived from the complete problem. Solve the reduced problem using a similar (nested) approach. Use this result to improve the current solution to the complete problem.

D. OBJECTIVE AND CONTENTS OF THE THESIS

One goal of this thesis is to investigate the advances made so far in this field, in particular, the application of the multigrid philosophy to the solution of long transportation problems. Previous work shows some success in obtaining approximate solutions to the problem, but on large problems the results are not very promising. This research steps back to a more general and abstract level in an attempt to find characteristics helping to define problems amenable to multigrid approach. Also, some of the essential concepts characterizing multigrid are generalized to a higher level of abstraction in order to seek new interpretations for the multigrid techniques when considered in the special context of optimization.

It is an objective of this thesis to gather what we could call a set of fundamental "building blocks". Multigrid is not a developed technique in optimization. Nested iteration and coarse grid correction require special handling in an optimization environment. We wish to review some of those optimization techniques already available that can be exploited, particularly those likely to be more adaptable to a multigrid use. Multilevel methods are a logical choice, so we examine and include them in the multigrid context.

An important feature of real-life network problems, including transportation problems, is the arc capacity. In the multigrid context, capacity has a special importance, since it tends to make the behavior of networks less smooth in response to iterative processes, causing multigrid implementation to be more problematic. We want to pay special attention to arc capacity and, whenever possible, include it as a factor in the problem design. This is an aspect that has not been considered before.

Finally, this work will try to open the interpretation of the multigrid idea in optimization to possible ways of restriction, other than the "traditional" node aggregation. This involves an abstraction of the concept of "grid". Also, possible ways to improve the relaxation process are suggested using the mentioned building blocks.

Chapter II is a description of the transportation problem. This type of problem has been historically selected to investigate new algorithms. This is mainly due to its simplicity in structure. In this chapter we present the transportation problem as a member of the class of minimum cost problems. Duality, a key concept in optimization theory that will be found along the pages to follow, is introduced. Specifically, we study the meaning of the dual associated to the transportation problem.

Although the long transportation problem is our selected problem type, a general approach to the solution of optimization (network) problems is kept in mind at all time. Occasionally, we will be willing to treat a more general type of problem.

The optimality conditions in networks are the basic set of tools which are constantly invoked in this work. It is natural then to devote some space to set up these conditions in networks. That is done in Chapter III, where it is preceded by a short summary of the more general linear programming optimality conditions.

Chapters IV, V, and VI describe the above mentioned building blocks. These chapters describe with some detail the decomposition methodology, the aggregation/disaggregation method, and the more recent scaling techniques, respectively. We describe them and give some examples. This is especially detailed in the chapter devoted to scaling. This last technique, dated 1972 (Edmonds and Karp), has been more extensively researched in recent years (Bertsekas, 1979; Bland and Jensen, 1992, Ahuja *et al*, 1993), and appears to be promising.

Chapter VII studies the perturbation method. It is an application of the more general perturbation philosophy to solving linear programming problems.

In Chapter VIII some further work in the multigrid concept is described. A certain familiarity with the method is assumed on the part of the reader. This work adds some insights to the comprehension of the multigrid methodology.

Chapter IX develops some ideas to identify the type of optimization problems that could be expected to be solved using multigrid. A short overview of previous work on the topic is presented, with conclusions about critical aspects of the implementation of multigrid algorithms in solving optimization problems, and suggestions for further research.

II. OPTIMALITY CONDITIONS

A. INTRODUCTION

In this chapter we present the Karush-Kuhn-Tucker (KKT) optimality conditions. These are necessary and sufficient for linear programming problems. The case of equality constraints is specifically considered. A key result in establishing the KKT conditions is the lemma presented in section B. Finally, in section D we specialize the optimality conditions to the case of networks.

A ***convex cone*** generated by the vectors $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(p)}$ is (Owen, 1982) the set of all vectors $\mathbf{v}$ such that

$$v = \sum_{k=1}^{p} \lambda_k v^{(k)}$$

for some nonnegative $\lambda_1, \lambda_2, \ldots, \lambda_p$.

NOTE *(Special notation)*: In this chapter we will use *dot-notation* to refer to sets defined using two indices, in the following way: to group elements indexed (i,j), with a constant value of i, and all possible values for j we use the special notation "i." (the dot is part of the notation). So, the ith row of a matrix $\mathbf{A}$ is denoted $A_{i.}$, while the jth column of $\mathbf{A}$ is denoted as $A_{.j}$.

B. LEMMA (Farkas' Lemma) :

One and only one of the following two systems has a solution:

$$\textit{System 1:} \quad \mathbf{w}\,\mathbf{A} = \mathbf{c}, \quad \mathbf{w} \geq \mathbf{0}$$

$$\textit{System 2:} \quad \mathbf{A}\,\mathbf{x} \geq \mathbf{0}, \quad \mathbf{c}\,\mathbf{x} < \mathbf{0} \qquad \text{(II.1)}$$

where A is a given *m x n* matrix and c is a given *n*-vector (a row vector).

(NOTE: There are other versions of the lemma; this is taken from Bazaraa, Jarvis and Sherali, 1990). We do not include a proof, but a geometric interpretation.

Geometric Interpretation: System 1 has a solution if and only if **c** belongs to the convex cone generated by the rows of **A**. On the other side, *system 2* has a solution if and only if the vector **c** does not belong to the cone generated by the rows of **A**.

C. KARUSH-KUHN-TUCKER (KKT) OPTIMALITY CONDITIONS

We take this version of the KKT optimality conditions from Bazaraa, Jarvis and Sherali, 1990. Some definitions are required.

A nonempty set S in $\Re^n$ is said to be ***convex*** if the line segment joining any two points in the set also belongs to the set. This is expressed as

$$\lambda\,\mathbf{x} + (1 - \lambda)\,\mathbf{y} \in S, \text{ for all } 0 \leq \lambda \leq 1 \text{ and all } \mathbf{x}, \mathbf{y} \in S$$

A real-valued function f defined on such set S is said to be a ***convex function*** if for every pair $\mathbf{x}, \mathbf{y} \in S$ and for every real number λ contained in the open interval (0,1), the following holds

$$f(\lambda\,\mathbf{x} + (1 - \lambda)\,\mathbf{y}) \leq \lambda\, f(\mathbf{x}) + (1 - \lambda)\, f(\mathbf{y})$$

Let $\bar{\mathbf{x}}$ be a member of a convex set S. A nonzero vector **d** is called a ***feasible direction*** at $\bar{\mathbf{x}}$ if there exists a $\delta > 0$, such that for all λ in the open interval $(0, \delta)$, the point $\bar{\mathbf{x}} + \lambda \mathbf{d}$ is in the set S.

A nonzero vector **d** is called an **improving feasible direction** at $\bar{\mathbf{x}}$ if there exists a $\delta > 0$, such that for all λ in the open interval $(0, \delta)$:

(i) $\bar{\mathbf{x}} + \lambda \mathbf{d}$ is a member of the set S;

(ii) $\mathbf{c}(\bar{\mathbf{x}} + \lambda \mathbf{d}) < \mathbf{c}\bar{\mathbf{x}}$ (for minimization problems, a descent direction).

Note that in linear programming problems, the cost vector **c** represents the gradient of the objective function. Condition (ii) in this case, is equivalent to $\mathbf{c}\,\mathbf{d} < 0$.

Definition: We say that a constraint is *binding* at $\bar{\mathbf{x}}$ if it is satisfied with equality at that point.

Now consider the linear programming problem.

$$\begin{aligned} \min \quad & \mathbf{c}\,\mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\,\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

For the above linear programming problem let $\bar{\mathbf{x}}$ be a feasible solution and **G** the submatrix of **A** formed by the binding constraints at $\bar{\mathbf{x}}$. The rows of **G** can be one of the following two types:

(a) Rows of **A**, of the form A_i such that $A_i\bar{\mathbf{x}} = b_i$; the index set for these constraints will be denoted as I ;

(b) Rows formed by zeros, except for a '1' in the jth position. The index set for these will be called *J*. These constraints belong to the group of nonnegativity constraints in the linear programming problem.

In other words,

$$I = \{ i : A_{i.}\bar{x} = b_i \}$$

$$J = \{ j : \bar{x}_j = 0 \}$$

Now suppose that $\bar{\mathbf{x}}$ is an optimal solution. Then there cannot be at $\bar{\mathbf{x}}$ any improving feasible direction **d**. Thus, (i) and (ii) cannot be satisfied simultaneously for any direction **d**. So, it cannot be that $\mathbf{cd} < \mathbf{0}$ and $\mathbf{G}(\bar{\mathbf{x}} + \lambda\,\mathbf{d}) \geq \mathbf{g}$.

Let **g** be the vector defined by the binding constraints $\mathbf{G}\bar{\mathbf{x}} = \mathbf{g}$. The second inequality above can be expanded as

$\mathbf{G}\bar{\mathbf{x}} + \lambda\,\mathbf{G}\,\mathbf{d} \geq \mathbf{g}$, which is equivalent to $\mathbf{G}\,\mathbf{d} \geq \mathbf{0}$.

Since there cannot be a direction **d** such that $\mathbf{cd} < \mathbf{0}$ and $\mathbf{G}\,\mathbf{d} \geq \mathbf{0}$ hold simultaneously, we can make use of Farkas' Lemma to state that there exists a vector $\mathbf{u} \geq \mathbf{0}$ such that $\mathbf{uG} = \mathbf{c}$.

Let us write such vector as $\mathbf{u} = (\alpha, \beta)$, where

$$\alpha = (u_i : i \in I)$$

$$\beta = (u_j : j \in J)$$

then the conditions $\mathbf{u} \geq \mathbf{0}$ and $\mathbf{uG} = \mathbf{c}$ can be rewritten as

$$\sum_{i \in I} \alpha_i A_{i.} + \sum_{j \in J} \beta_j e_j = c \qquad \text{(II.2)}$$

with

$$\alpha_i \geq 0, \quad i \in I \quad \text{and} \quad \beta_j \geq 0, \quad j \in J \tag{II.3}$$

These are the KKT conditions for optimality. The interpretation is: $\bar{\mathbf{x}}$ is an optimal solution to the linear program above if and only if the objective gradient $\mathbf{c}$ lies in the convex cone generated by the gradients of the binding constraints at $\bar{\mathbf{x}}$.

These conditions are usually written in the following form. Define the vectors

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m) \geq 0$$

$$\beta = (\beta_1, \beta_2, \ldots, \beta_n) \geq 0$$

their components having value zero for those indices not belonging to *I*, *J*, respectively (recall that the lengths, m and n, are precisely the number of constraints and the number of variables of the problem). Those are indices corresponding to the nonbinding constraints. Then the KKT conditions hold for vectors ($\mathbf{x}$, α, β) if there exists a solution to the system

$$\mathbf{Ax} \geq \mathbf{b}, \qquad \mathbf{x} \geq \mathbf{0} \tag{II.4}$$

$$\alpha\mathbf{A} + \beta = \mathbf{c}, \qquad \alpha \geq 0, \quad \beta \geq 0 \tag{II.5}$$

$$\alpha(\mathbf{Ax} - \mathbf{b}) = 0, \quad \beta\mathbf{x} = 0 \tag{II.6}$$

The first condition ***(primal feasibility)*** states that the candidate point must be feasible. Condition (II.5) is the ***dual feasibility***. Here α and β are the ***Lagrangian***

multipliers or ***dual variables***. Finally, the last condition is the ***complementary slackness condition***.

The complementary slackness condition is the expression of the so called **Complementary Slackness Theorem**, stating that if a pair of primal and dual solutions are feasible and satisfy the complementary slackness condition, they are optimal.

An immediate consequence of the above is the following:

Corollary: The following two statements are equivalent

(i) $\bar{\mathbf{x}}$ is an optimal solution for the primal problem and $\bar{\mathbf{u}}$ is an optimal solution for the dual problem;

(ii) $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$, is a feasible solution for the pair of problems, primal and dual.

In the case of equality constraints, which concerns the transportation problem, we have:

$$\begin{aligned} &\min \quad \mathbf{c}\,\mathbf{x} \\ &\text{s.t.} \quad \mathbf{A}\,\mathbf{x} = \mathbf{b} \\ &\qquad \mathbf{x} \geq \mathbf{0} \end{aligned}$$

We can transform the equality into a logical combination of two inequalities,

$$(\mathbf{A}\,\mathbf{x} = \mathbf{b}) \equiv (\mathbf{A}\,\mathbf{x} \geq \mathbf{b}) \quad \text{and} \quad (-\mathbf{A}\,\mathbf{x} \geq -\mathbf{b}) \qquad \text{(II.7)}$$

As a result, the values of α are unrestricted at the points satisfying the KKT conditions, and these are expressed in the form:

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}, \qquad \mathbf{x} \geq \mathbf{0} \tag{II.8}$$

$$\alpha\,\mathbf{A} + \beta = \mathbf{c}, \qquad \alpha \text{ unrestricted}, \quad \beta \geq 0 \tag{II.9}$$

$$\beta\,\mathbf{x} = \mathbf{0} \tag{II.10}$$

Suppose $\bar{\mathbf{x}}$ is an optimal solution. Then, in order to satisfy the complementary slackness condition (II.9), it must be that

$$(\beta_B\,,\ \beta_N)\ (\bar{\mathbf{x}}_B\,,\ \bar{\mathbf{x}}_N) = \beta_B\,\bar{\mathbf{x}}_B + \beta_N\,\bar{\mathbf{x}}_N = 0 \tag{II.11}$$

where the vectors β and $\bar{\mathbf{x}}$ have been expressed in blocks of components associated to the basic and nonbasic variables (subscripts B, N respectively).

Now, since $\bar{\mathbf{x}}_N = \mathbf{0}$, it must be that $\beta_B\,\bar{\mathbf{x}}_B = \mathbf{0}$, and (II.11) holds when $\beta_B = 0$. Then (II.8) can be expressed:

$$(c_B,\ c_N) - \alpha(\mathbf{B},\ \mathbf{N}) - (\beta_B,\ \beta_N) = (0,\ 0) \tag{II.12}$$

equivalent to

$$(c_B - \alpha B = 0) \quad \text{and} \quad (c_N - \alpha N - \beta_N = 0) \tag{II.13}$$

where N is the submatrix of A formed by the rows associated to the nonbasic variables.

D. OPTIMALITY CONDITIONS IN NETWORKS

The theorems, properties and corollaries in this section can be found in Ahuja *et al*, 1993. Suppose **N** is a network. A set of node potentials is a mapping π: **N** -> **R**. Let $\pi = \{\pi(1), \pi(2), ..., \pi(n)\}$ be a set of node potentials. For compactness, sometimes we will treat π as a vector. We define the ***reduced cost*** of an arc (i,j) as

$$c_{ij}^{\pi} = c_{ij} - \pi(i) + \pi(j) \qquad \text{(II.14)}$$

We use the superscript to indicate that the reduced cost is associated with the set π. Note that when $\pi = \mathbf{0}$ the reduced cost vector is identical to the cost vector. (Also for compactness, we refer to both *cost sets* in terms of vectors, which we will do sometimes to simplify the notation).

Lemma: Suppose we want to solve a minimum cost flow problem using as set of costs the reduced costs associated with a set of node potentials π. Let $z(\pi)$ denote the value of the objective function value of this problem, and $z(0)$ the objective value considering the usual costs. The flow values are the same in both cases. Then we have

$$\sum_{i,j} c_{ij}^{\pi} = \sum_{i,j} c_{ij}x_{ij} - \sum_{i \in N} \pi(i)d(i) \qquad \text{II.15)}$$

or

$$z(\pi) = z(0) - \pi\, \mathbf{d} \qquad \text{(II.16)}$$

where **d** is the vector of demands and π the vector of node-potentials.

Proof: Set all node potentials to zero. Increasing the potential of node k from 0 to $\pi(k)$ modifies the reduced costs as follows:

- all incoming arcs to node k increase reduced costs by $\pi(k)$;

- all arcs leaving k decrease reduced costs by $\pi(k)$.

This causes the value of the objective function to decrease to:

$$z(\pi_k) = z(0) + \pi(k)\sum_i x_{ik} - \pi(k)\sum_j x_{kj}$$
$$= z(0) - \pi(k)d(k) \qquad \text{(II.17)}$$

Proceeding as above for all the nodes yields (II.16)

Corollary (II-1): Optimal minimum cost flow problems with arc costs c_{ij} or c^{π}_{ij} have the same flow distributions.

Corollary (II-2): For any directed cycle C and for any set of node potentials π:

$$\sum_C c^{\pi}_{ij} = \sum_C c_{ij} \qquad \text{(II.18)}$$

Corollary (II-3): For any directed path P from k to l :

$$\sum c^{\pi}_{ij} = \sum c_{ij} - \pi(k) + \pi(l) \qquad \text{(II.19)}$$

Let **u** represent the vector of arcs capacities. We define the ***residual network*** with respect to a given flow $\bar{\mathbf{x}}$ as follows: replace each arc (i,j) in the original network by two arcs (i,j) and (j,i):

(i) the arc (i,j) has cost c_{ij} and *residual capacity* $r_{ij} = u_{ij} - \bar{x}_{ij}$ and

(ii) the arc (j,i) has cost $-c_{ij}$ and residual capacity $r_{ji} = \bar{x}$.

The residual network consists of only the arcs with a positive residual capacity.

The following theorems (Ahuja *et al*, 1993) state the optimality conditions in network problems (for brevity, the proofs have not been included here).

Theorem II-1 (Negative Cycle Optimality Conditions): A feasible solution $\bar{\mathbf{x}}$ is an optimal solution of the minimum cost problem if and only if the residual network $G(\bar{\mathbf{x}})$ contains no negative cost (directed) cycle.

Theorem II-2 (Reduced Cost Optimality Conditions): A feasible solution $\bar{\mathbf{x}}$ is an optimal solution of the minimum cost problem if and only if some set of node potentials π satisfy $c_{ij}^{\pi} \geq 0$ for every arc (i,j) in the residual network $G(\bar{\mathbf{x}})$.

Theorem II-3 (Complementary Slackness Optimality Conditions): A feasible solution $\bar{\mathbf{x}}$ is an optimal solution of the minimum cost problem if and only if for some set of node potentials π, the reduced costs and flow values satisfy the following for every arc (i,j) in the network:

(i) If $c_{ij}^{\pi} > 0$, then $\bar{x}_{ij} = 0$ (II.20)

(ii) If $0 < \bar{x}_{ij} < u_{ij}$ then $c_{ij}^{\pi} = 0$ (II.21)

(iii) If $c_{ij}^{\pi} = 0$, then $\bar{x}_{ij} = u_{ij}$ (II.22)

The above sections provide a method for checking for optimality of solutions in optimization problems, and its specialization to network flows. This will be useful when revising particular techniques in the chapters to follow.

III. THE TRANSPORTATION PROBLEM IN THE CONTEXT OF MINIMUM COST PROBLEMS

A. INTRODUCTION

The transportation problem was first introduced by Hitchcock in 1941, and since then a wide variety of seemingly different problems have been shown to be equivalent to the transportation problem. Historically, the transportation model has served as "laboratory animal" in the development of optimization technology (Dantzig, 1963). No matter how large the problem at hand, it possesses a simple, exploitable structure.

The basic structure is the following: a company owns m warehouses (***origins***, or ***supply nodes***), in each of which there is a given amount of a certain commodity in stock. Also, there are n retailers (***destinations*** or ***demand nodes***), each with a given demand for this commodity. The unit transportation cost from each warehouse to every retailer is known data. The ***objective*** is to ship units from supply nodes to demand nodes, such that:

(i) the total shipping cost is minimized;

(ii) the demands of retailers are satisfied;

(iii) no more units leave a warehouse than there are in stock;

If the total supply equals the total demand we have a ***balanced*** transportation problem. Actually, we can assume that all transportation problems can be formulated as balanced, adding dummy supply (demand) nodes to provide (absorb) the needed (extra) supply (demand).

When the number of supply nodes is small compared with the number of demand nodes we have a ***long*** transportation problem.

When there are limits imposed on the capacity of the supply from the different origins to the different destinations we have a ***capacitated*** transportation problem. This is a more interesting problem, for practical purposes. In this work we will focus on the simplest of the models, defined above, with many more demand nodes than supply nodes (long version).

The formulation of a transportation problem involving 'm' origins and 'n' destinations is:

$$\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{III.1a}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = s_i, \qquad i=1,2,...,m \tag{III.1b}$$

$$\sum_{i=1}^{m} x_{ij} = d_j, \qquad j=1,2,...,n \tag{III.1c}$$

$$x_{ij} \geq 0, \qquad i=1,...,m; \quad j=1,...,n \tag{III.1d}$$

where

c_{ij} is the cost per unit for shipping from i to j;

x_{ij} is the number of units shipped from i to j;

s_i is the available supply at node i;

d_j is the demand required by node j.

The above model has $m \cdot n$ decision variables x_{ij}. The set of constraints defined in (III.1b-c) is written $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ in compact form, and is composed of $m + n$ constraints.

The matrix A has one row for each node, and one column for each arc (i,j) joining supply-demand nodes. Furthermore, each column contains two non-zero coefficients, and they are precisely +1 and -1 (there is another version, called "Generalized Transportation Problem", which allows for coefficients other than zero and one in the demand constraints). The rank of A is m+n-1. The matrix A is ***totally unimodular***, i.e. the determinant of every square submatrix formed from it has value -1, 0, or 1. This fact allows for integer optimal solutions when both supplies and demands are integer values (Bazaraa *et al...*, 1990)

B. TRANSPORTATION TABLEAU

A convenient form of representing the transportation model is by means of a tableau where the rows 1, 2, ... , m represent origins, and the columns 1, 2, ..., n represent destinations. Each cell (i,j) contains the flow variable x_{ij} and/or the cost c_{ij} (Figure 1). This representation is commonly used for describing particular strategies of solution to the transportation problem. This will be seen particularly in Chapter V. Here it is presented for compactness.

C. GRAPH AND NETWORK REPRESENTATION

The underlying structure in a transportation problem is that of a directed network. More specifically, it is a complete bipartite graph, i.e. the set of nodes is partitioned into two subsets: origins and destinations; every origin is connected to every destination (and

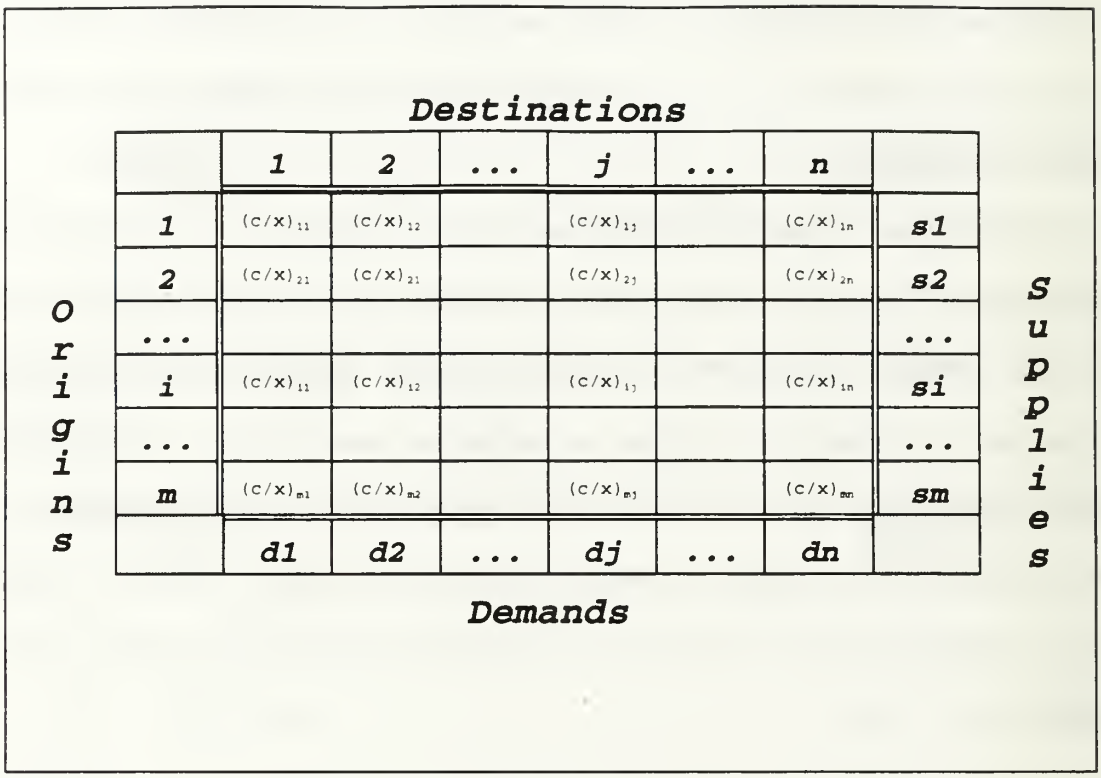

Figure 1.- Transportation Tableau

vice-versa); finally, all the arcs defining the problem go from a supply node to a demand node. In graph terminology, every origin is *adjacent to* all destinations and no destination is adjacent to any origin. (Note that a pair of supply-demand nodes not adjacent to each other are joined by an arc highly penalized in cost). The schematic representation is in Figure 2.

In order to complete the network representation of the graph we need to add an additional node. The resulting network has 'm+n+1' nodes. This translates into the linear programming model as a new ***artificial*** variable. We do not need to put costs in the originated arcs, since the artificial variable value must be zero in any feasible solution (Bazaraa, 1990). The additional node plays the role of a ***root node***.

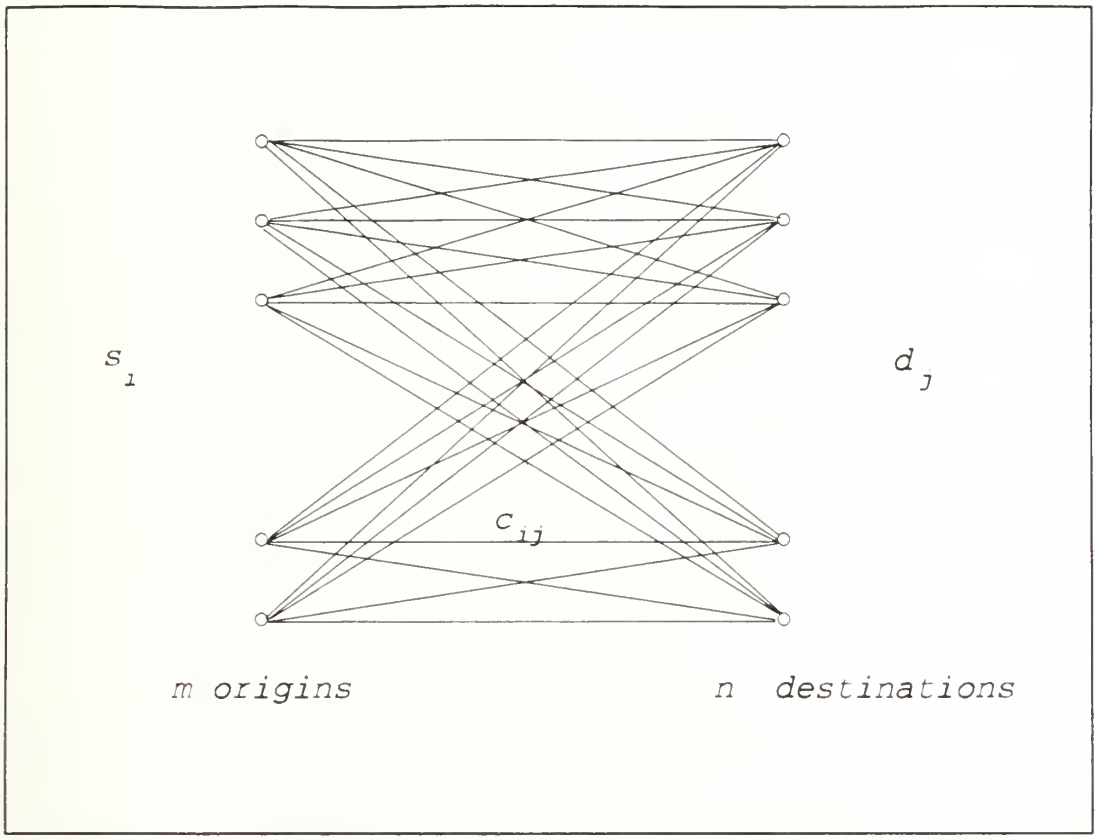

Figure 2.- ***Bipartite Graph associated with a Transportation Problem.***

A basis is now represented by a rooted spanning tree (i.e. a spanning tree plus an extra arc, ***root arc***, corresponding to the artificial variable). Figure 3 represents the associated network with the transportation problem, and a basic feasible solution.

Since every destination must have its demand satisfied, the solutions to a transportation problem have at least one positive flow arriving to each demand node. This is of special interest in long transportation models. In those models, at optimality the total number of positive flows is at most $m+n-1$. If so, then we have that at most $(m+n-1) - n = m - 1$ demand nodes will be supplied by two or more supply nodes while the remaining at least $n - m + 1$ demand nodes will be supplied by only one supply node.

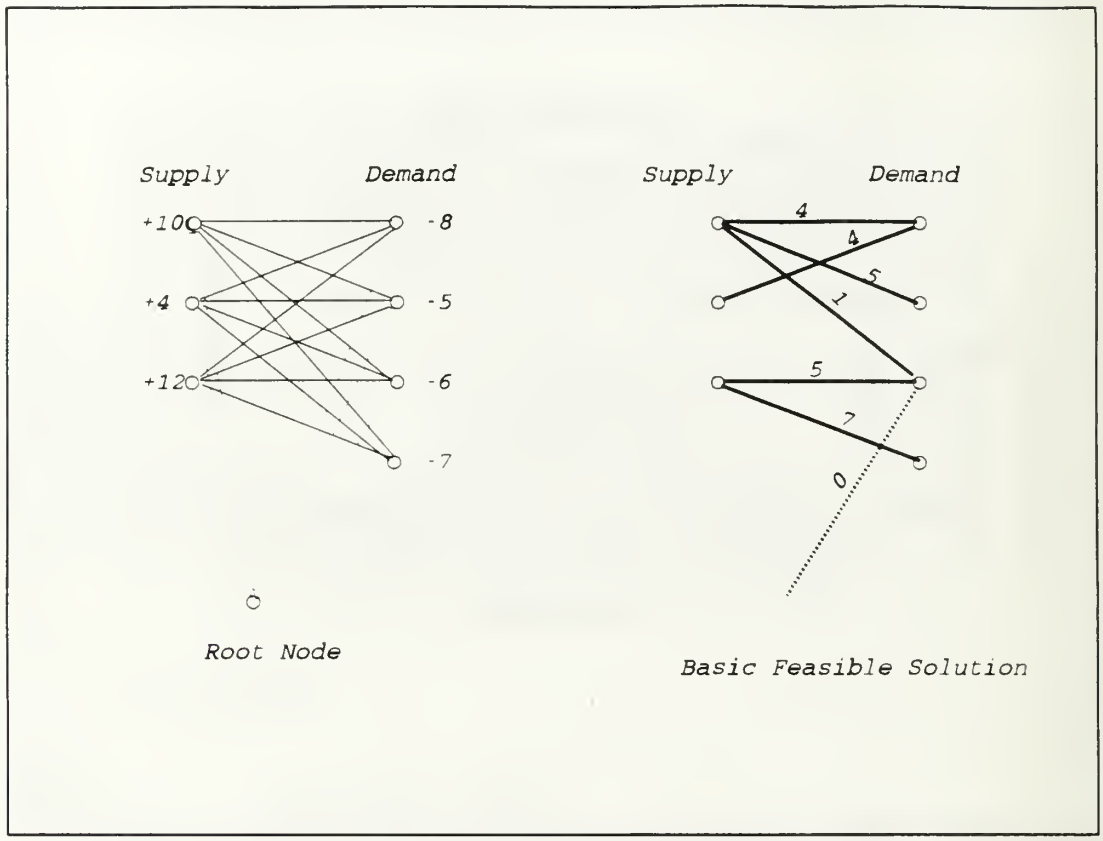

Figure 3.- ***Transportation Problem. Network Representation and Basic Feasible Solution.***

D. DUAL PROBLEM

It is interesting to formulate the dual of the transportation problem. Recall that in dual problems (Bazaraa *et al*, 1990), there is a dual variable corresponding to each constraint. In the present case, we establish two groups of dual variables: those associated with constraints (III.1b) are designated u_i, and those associated with constraints (III.1c) are designated v_j. The dual problem is stated

$$\max \quad \sum_{i=1}^{m} s_i u_i + \sum_{j=1}^{n} d_j v_j \qquad \text{(III.2a)}$$

$$\text{s.t.} \quad u_i + v_j + c_{ij}^{*} = c_{ij} \qquad \text{(III.2b)}$$

$$u_i \,, \quad v_j \quad \text{unrestricted} \qquad \text{(III.2c)}$$

The slack variables c_{ij}^{*} are introduced in (III.2b) to obtain equality.

The primal-dual relationship can be explained as follows: suppose that a firm Alpha has to transport goods as defined by the primal problem. The firm is planning to hire a smaller firm, Beta, to do the job instead. The hiring conditions must be such that the costs claimed by Beta to Alpha can never be bigger than those defined by the primal setting (otherwise, Alpha would be willing to do the job). So Alpha puts up the following proposal: Beta will receive u_i dollars for each unit shipped out of the i-th origin. Beta also will receive v_j dollars for each unit delivered to the j-th destination (III.2a). Constraints (III.2b) state that the shipment of a unit from i to j cannot be greater than the shipment cost of a unit in the primal problem (***Weak Duality Property***). Within this setting, Alpha sets Beta free to arrange the transportation policy, And Beta will try to maximize profits. In doing so, the best profit (optimal objective function value) that Beta can produce is exactly the least expensive (optimal) solution that Alpha could come to by itself (***Strong Duality Property***).

Suppose that we have a feasible solution **x**. Since (any) one of the primal constraints is redundant, (any) one of the dual variables can assume an arbitrary value.

One way to check for optimality of **x** is by checking whether the primal feasibility and dual feasibility are accomplished. This is known as the ***supervisor's principle***. That is, if **u**, **v** are the dual vectors (variables) associated with **x**, the fact that **u**, **v** are feasible for the dual problem, automatically implies that **x** is an optimal solution for the *primal* problem.

E. OPTIMALITY

In the transportation problem the rank of the matrix A is m+n-1. To apply the simplex method we need full rank. We solve that with an artificial variable (Bazaraa, 1990). Then,

any basis will consist of m+n-1 variables, plus an artificial variable. In feasible solutions the value of this artificial variable is zero.

Recall the primal (III.1a-d) and the dual (III-2a-c) of the transportation problem. Recall also that the Complementary Slackness Theorem states that at optimality, if a variable is positive, then the corresponding constraint in the other problem must be tight. And if a constraint in one problem is not tight, then the corresponding variable in the other problem must be zero.

The process of finding an optimal solution should follow the next general steps:

(i) Start with a feasible solution **x**;

(ii) Find the duals (**w**, **v**) associated with **x**;

(iii) Check if the dual variables are a feasible solution to the dual problem, or price out the nonbasic variables and determine if they price out unfavourably (i.e. they are not eligible to enter the basis).

In the described process, the duals can be computed as solution to the system:

$$[w \quad v]\ B = c_B^T \tag{III.3}$$

and the pricing operation of the nonbasic variable 'k' is performed by computing:

$$c_k - [w \quad v]\ N_{.k} \tag{III.4}$$

the special structure of the problem makes (III.4) easy to compute, since

$$c_{ij} - [w \quad v]\ N_k = c_{ij} - (w_i - v_j) \tag{III.5}$$

The previous facts are easily followed using the network structure of the problem. Algorithms like the simplex network and GNET use them very efficiently. The simplex method maintains a basic feasible solution at every step. Given a basic feasible solution, the method first applies the optimality criteria to test the optimality of the current solution.

If the current solution does not fulfill this condition, the algorithm performs an operation, known as *pivoting*, to obtain another basis structure with a lower(minimization) or identical cost. The simplex method repeats this process until the current basic feasible solution satisfies the optimality criteria. GNET (Bradley *et al* , 1977) is a network simplex algorithm that uses the idea of basic trees described previously in this chapter, together with extremely efficient data structures to represent the necessary information about the problem. For a description of network simplex algorithms, see Ahuja, *et al* , 1993.

In the next three chapters, we consider some of the multilevel methods existing in the optimization literature.

IV. THE DECOMPOSITION METHOD

A. INTRODUCTION

"Decomposition" is a term that embraces three equivalent procedures for dealing with large linear programming problems. The procedures are the Dantzig-Wolfe Decomposition, Benders' Decomposition and the Lagrangian Relaxation. The usual environment is that of a problem having a special set of constraints (hopefully very easy to handle), together with another set of *bundle* constraints that make the problem harder.

The three methods involve systematic computation at two levels. One level is called ***The Master Problem***. The other is called ***The Subproblem***. The subproblem works on the easy set of constraints. Normally, the subproblem is solved quickly. But its solution does not necesarily satisfy the bundle (harder) constraints. The solution of the subproblem provides information via feedback to the master problem, specifically in the problem parameters. The modified master problem originates a new subproblem. The control is passed again to the subproblem level, where a new solution is produced. The process continues until it converges and an optimal solution is found satisfying the bundle constraints. The following sections will cover an overall description of the various techniques. The goal is to broaden the spectrum of multilevel techniques that could be used to produce multigrid approaches to optimization problems.

In Appendix A, a numerical example is provided showing a practical application of the Dantzig-Wolfe decomposition scheme.

B. DANTZIG-WOLFE DECOMPOSITION

Consider the following linear program:

$$\min \quad cx \tag{IV.1a}$$

$$\text{s.t.} \quad Ax = b \tag{IV.1b}$$

$$x \in X \tag{IV.1c}$$

Here, (IV.1c) is the set defined by special constraints; (IV.1b) represents the bundle constraints. If X is bounded and polyhedral, then any point in X can be expressed as

$$x = \sum_{j=1}^{t} \lambda_j x_{(j)} = x(\lambda)$$

with $$\sum_{j=1}^{t} \lambda = 1$$

and $$\lambda_j \geq 0; \qquad j = 1,2,...,t$$

where $\mathbf{x}_{(j)}$ are the t extreme (corner) points of the set X.

Substituting in (IV.1), the original problem is now expressed in terms of the new variables λ_i as follows:

$$\min \quad \sum_{j=1}^{t} (cx_{(j)})\ \lambda_j \tag{IV.2a}$$

$$\text{s.t.} \quad \sum_{j=1}^{t} (Ax_{(j)})\ \lambda_j = b \qquad (w) \tag{IV.2b}$$

$$\sum_{j=1}^{t} \lambda_j = 1 \qquad (\alpha) \tag{IV.2c}$$

This is called the ***Master Problem***, in the usual terminology. The variables **w** and α in parentheses are the dual variables associated with each respective set of constraints of the master problem. So **w** actually is a vector of length equal to the number of rows in A, whereas α is a single variable, since (IV.2c) defines only one constraint.

At this point, it is useful to write the dual problem associated with the master problem:

$$\max \quad \mathbf{w}\,\mathbf{b} + \mathbf{v}\,\mathbf{d} \tag{IV.3a}$$

$$\text{s.t.} \quad \mathbf{w}\,(\mathbf{A}\,\mathbf{x}_{(j)}) + \alpha \leq (\mathbf{c}\,\mathbf{x}_{(j)}) \tag{IV.3b}$$

$$\mathbf{w},\ \alpha \quad \text{unrestricted} \tag{IV.3c}$$

Referring back to the Master Problem, trying to solve (IV.2) is a hard task, since the number of variables is normally very large. They correspond to the different corner points of X, and finding them is computationally expensive, perhaps prohibitively so.

Now consider the following problem:

$$\max \quad (\mathbf{w}\,\mathbf{A} - \mathbf{c})\,\mathbf{x} \tag{IV.4a}$$

$$\text{s.t.} \quad \mathbf{x} \in X \tag{IV.4b}$$

Note that this problem is bounded and not empty, so it attains an optimal solution at one corner point of X. Note also that the solution solves precisely the pricing operation that takes place when applying the Simplex method to (IV.2):

$$\max \{z_j - c_j\} = \max \left\{ [w,\ \alpha] \begin{bmatrix} (Ax_{(j)}) \\ 1 \end{bmatrix} - cx_{(j)} \right\} \tag{IV.5}$$

where the index runs over 1,2,...,t, the corner points of X. The set (IV.4) defines the ***Subproblem***, and is easily solved because of the special structure of its constraints. Actually, the *Subproblem* task in the whole context is to take advantage of the special structure of the constraints defining the set X in order to solve the pricing operation of the *Master Problem* as a fast linear program.

In summary, the Dantzig-Wolfe method proceeds in the following way:

(i) Start with a feasible solution for the master problem (IV.1);

(ii) Solve the associated subproblem;

(iii) The solution to the subproblem provides the coordinates of a new corner point that comes into play;

(iv) The subproblem also provides an entering variable in the basis of the Simplex Tableau. After pivoting, the basis inverse, dual variables, and RHS are updated;

(v) Now we have an improving basic feasible solution of the master problem. The set of λ's provide a new convex linear combination which is a feasible solution (the best so far) to the original problem (IV.1);

(vi) Iterate the process to convergence of the solution.

Finally, it can be shown (Bazaraa *et al*, 1990) that, at each iteration, we obtain an upper and lower bound to the optimal objective function of (IV.1). So the computations can be stopped at a determined level of accuracy of the current solution.

A numerical example of the practical use of this method is given in Appendix B.

C. BENDERS' DECOMPOSITION

Benders' decomposition scheme performs a complementary procedure to that in Dantzig's method. Here, the dual is really the problem to solve. But let us start with the primal. Consider the following problem:

$$\min \quad \mathbf{c}\,\mathbf{x} \tag{IV.6a}$$

$$\text{s.t.} \quad \mathbf{A}\,\mathbf{x} = \mathbf{b} \qquad (\mathbf{w}) \tag{IV.6b}$$

$$\mathbf{D}\,\mathbf{x} \geq \mathbf{d} \qquad (\mathbf{v}) \tag{IV.6c}$$

The dual to this problem is

$$\max \quad \mathbf{w}\,\mathbf{b} + \mathbf{v}\,\mathbf{d} \tag{IV.7a}$$

$$\text{s.t.} \quad \mathbf{w}\,\mathbf{A} + \mathbf{v}\,\mathbf{D} \leq \mathbf{c} \tag{IV.7b}$$

$$\mathbf{w} \text{ unrestricted}, \quad \mathbf{v} \geq 0 \tag{IV.7c}$$

This is the problem to be solved using Benders' method. It is interesting to keep considering it as dual, in order to view it in terms of a decomposition scheme.

Consider $\mathbf{w}$ as a parameter. Then restate (IV.7) as:

$$\max_{(\mathbf{w}\ \text{unrestricted})} \left\{ \mathbf{w}\mathbf{b} + \max_{\substack{\text{s.t. } \mathbf{v}\mathbf{D} \leq \mathbf{c} - \mathbf{w}\mathbf{A} \\ \mathbf{v} \geq 0}} (\mathbf{v}\mathbf{d}) \right\} \tag{IV.8}$$

Note that the dual of the inner maximization problem in (IV.8) is:

$$\left\{ \begin{array}{l} \min \ (\mathbf{c} - \mathbf{w}\mathbf{A})\,\mathbf{x} \\ \text{s.t.} \ \mathbf{D}\mathbf{x} \geq \mathbf{d} \\ \qquad \mathbf{x} \geq 0 \end{array} \right\} \tag{IV.9}$$

Therefore, at optimality, the objective function of (IV.9) is equivalent to that of the mentioned inner problem. This fact allows us to express (IV.8) as:

$$\max_{(w \text{ unrestricted})} \left\{ wb + \min_{s.t.\ x \in X} (c - wA)x \right\} \qquad \text{(IV.10)}$$

Since the current inner minimization problem is bounded and nonempty, it attains an optimal solution at one of the extreme points of X. Suppose that X has t of these extreme points, and name them in similar fashion as we did in section A. Then we can formulate the following statement, equivalent to (IV.10):

$$\max_{(w \text{ unrestricted})} \left\{ wb + \min_{j = 1,2,...,t} (c - Wa)x_{(j)} \right\} = z \qquad \text{(IV.11)}$$

The last problem can be restated in the following terms:

$$\max \ z \qquad \text{(IV.12a)}$$

$$\text{s.t.} \quad z \leq wb + (c - wA)\, x_{(j)}, \qquad j = 1,2,...,t \qquad \text{(IV.12b)}$$

$$z, \ w \ \text{unrestricted} \qquad \text{(IV.12c)}$$

.

The above problem (IV.12) represents the ***Master Problem*** in this procedure. This setting (IV.12) expresses the previous problem (IV.11) without the two "levels". On the other hand (and this complicates things) the problem defined in (IV.12) has t constraints, one per corner point. Those are, in practice, too many constraints.

Let us underline the parallelism between this situation and the situation at stage (IV.3) in the Dantzig-Wolfe procedure. In the earlier case, the situation was handled by a ***column-generating*** procedure. Here we will proceed on a ***row-generating*** fashion. We will come back to this point.

It is useful to form the dual of (IV.12). To facilitate things, express (IV.12) as follows:

$$\max \quad z \tag{IV.13a}$$

$$\text{s.t.} \quad w(Ax_{(j)} - b) + z \leq cx_{(j)} \qquad (\lambda) \tag{IV.13b}$$

$$w, z \text{ unrestricted} \tag{IV.13c}$$

Again, we have written the next dual variables associated with the constraints in the problem. Next we can formulate the dual of (IV.13) to obtain:

$$\min \quad \sum_{j=1}^{t} (cx_{(j)})\ \lambda_j \tag{IV.14a}$$

$$\text{s.t.} \quad \sum_{j=1}^{t} (Ax_{(j)} - b)\ \lambda_j = 0 \tag{IV.14b}$$

$$\sum_{j=1}^{t} \lambda_j = 1 \tag{IV.14c}$$

Note that the last problem is precisely the Dantzig-Wolfe approach to (IV.6), where the corner points correspond to the set X defined by the constraints (IV.6c).

A way of alleviating the difficulty of dealing with a large number of constraints, that arose at (IV.12) is by a ***relaxation*** technique. It is based on the following fact: if we solve (IV.13) to optimality, but using only a subset of the constraint ′.13b), then the optimal solution, i.e. $(\bar{w}, \bar{z})$, attains a value of the objective function that is an upper bound for the original problem (IV.2). (It is easy to see that it is so, because the set of values of x defined by the subset of constraints contains the *feasible region* defined by (IV.12b), and both objective functions are the same). So if the pair $(\bar{w}, \bar{z})$ found is such that

it is feasible for the constraints (IV.13b) then the solution is optimal to the original problem. Again, this is a hard task, due to the large number of constraints.

The above check is equivalent to checking that

$$\bar{z} \leq \bar{w}b + \min_{x \in X} \{ (c - \bar{w}A)x \} \qquad \text{(IV.15)}$$

Problem (IV.15) is called ***The Subproblem*** in this decomposition.

Summarizing Benders' procedure:

(i) Solve the LP subproblem (IV.15).

(ii) If the solution satisfies (IV.15) then we are done.

(iii) Otherwise, let x_k be the optimal solution to (i). Add the constraint $z \leq wb + (c - wA)x_{(k)}$ to the relaxed problem (i).

(iv) Reoptimize until some relaxed problem gives an objective value $\bar{z}$ equal to the optimal value in (IV.15)

D. LAGRANGIAN RELAXATION

Lagrangian relaxation uses the idea of relaxing the explicit (bundle) constraints by bringing them into the objective function with associated Lagrange multipliers **w**.

So if our original problem is of the form (IV.1), then this technique relaxes the problem, restating it as follows:

$$\min \quad cx + w(Ax - b) \qquad \text{(IV.16)}$$

$$\text{s.t.} \quad x \in X \qquad \text{(IV.17)}$$

The resulting problem is called the ***Lagrangian Relaxation***, or ***Lagrangian Subproblem*** of the original problem.

If we define

$$L(w) = \text{Min} \{ cx + w(Ax - b) \} \qquad \text{(IV.18)}$$

as the Lagrangian Function, then the following holds:

Lemma: For any vector **w** of the Lagrangian multipliers, the value L(**w**) is a lower bound on the optimal objective function of the original optimization problem.

The proof can be written in abreviated fashion as:

$$\min \{ \mathbf{cx} : \mathbf{A}\,\mathbf{x} = \mathbf{b},\ \mathbf{x} \in X \}$$

$$= \min \{ \mathbf{c}\,\mathbf{x} + \mathbf{w}\,(\mathbf{A}\,\mathbf{x} - \mathbf{b}) : \mathbf{A}\,\mathbf{x} = \mathbf{b},\ \mathbf{x} \in X \}$$

$$\geq \min \{ \mathbf{c}\,\mathbf{x} + \mathbf{w}\,(\mathbf{A}\,\mathbf{x} - \mathbf{b}) : \mathbf{x} \in X \}$$

The above inequality is produced when the elimination of the constraints $Ax = b$ from the right hand side above cannot cause an increase in the value of the objective function (most likely the value will decrease).

Now, if we want the sharpest lower bound on the optimal objective value, we would need to solve the following optimization problem:

$$L^* = \max_{w} L(w) \qquad \text{(IV.19)}$$

which is called ***Lagrangian Multiplier Problem*** (or ***Lagrangian Dual***) associated with the original optimization problem.

Theorem: The optimal objective function value L^* of the Lagrangian multiplier problem is always a lower bound on the optimal objective function value of the original problem.

Proof: Let LS be the set of all the optimal objective values L(w). The maximum is a member of the set. Therefore it is also a lower bound for the optimal objective of the original problem.

So we have the following relationship:

$$L(w) \leq L^* \leq z^* = cx \qquad \text{(IV.20)}$$

The above leads to the following two properties:

Property (a) (Optimality test): Suppose that **w** is a vector of Lagrangian multipliers and **x** is a feasible solution to the optimization problem (IV.1) satisfying the condition $L(w) = cx$. Then L(**w**) is an optimal solution of the Lagrangian Multiplier Problem and **x** is an optimal solution of the optimization problem (IV.1).

Property (b): If for some choice of the Lagrangian multiplier vector **w** the solution **x** of the Lagrangian Relaxation is feasible in the optimization problem (IV.1), then **x** is an optimal solution to (IV.1) and **w** is an optimal solution to the Lagrangian Multiplier Problem.

Hence the Lagrangian relaxation gives a lower bound to the optimal value for the objective function of optimization problems. This is precisely its primary use.

Theorem: Suppose that we apply the Lagrangian Relaxation technique to a linear program (LP) defined as follows:

$$\begin{aligned} \min \quad & \mathbf{cx} \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \qquad (\mathbf{w}) \\ & \mathbf{Dx} = \mathbf{q} \qquad (\mathbf{v}) \\ & \mathbf{x} \geq \mathbf{0} \qquad (\beta) \end{aligned}$$

by relaxing the constraints $\mathbf{Ax} = \mathbf{b}$. Then the optimal value L^* of the Lagrangian multiplier problem **is equal to** the optimal objective function value of (LP).

Note the associated dual variables $\mathbf{w}$, $\mathbf{v}$. Suppose (w^*, v^*, β^*) are optimal solution for the dual problem. Let also $\mathbf{x}^*$ be the corresponding primal solution. By linear programming theory $\mathbf{x}^*, \mathbf{w}^*, \mathbf{v}^*, \beta^*$ satisfy the dual feasibility and complementary slackness conditions:

$$w^*\mathbf{A} + v^*\mathbf{D} + \beta^* = \mathbf{c} \qquad \text{(IV.21a)}$$

$$\beta^* x^* = \mathbf{0} \qquad \text{(IV.21b)}$$

$$\beta^* \geq \mathbf{0} \qquad \text{(IV.21c)}$$

Now, let us write the Lagrangian relaxation applied to (LP) for $L(\mu) = L(w^*)$:

$$\min \quad \mathbf{cx} - \mathbf{w}^*(\mathbf{Ax} - \mathbf{b}) \equiv (\mathbf{c} - \mathbf{w}^*\mathbf{A})\,\mathbf{x} + \mathbf{w}\,\mathbf{b} \qquad \text{(IV.22a)}$$

$$\text{s.t.} \quad \mathbf{D}\,\mathbf{x} = \mathbf{q} \qquad (\mathbf{v}) \qquad \text{(IV.22b)}$$

$$\mathbf{v} \geq \mathbf{0} \qquad (\beta) \qquad \text{(IV.22c)}$$

The dual feasibility and complementary slackness conditions for this problem are:

$$\bar{v}D + \bar{\beta} = c - w^{*}A \qquad \text{(IV.23a)}$$

$$\beta^{*}\bar{x} = 0 \qquad \text{(IV.23b)}$$

$$\bar{\beta} \geq 0 \qquad \text{(IV.23c)}$$

The solution $\mathbf{x}^{*}$ is feasible for the Lagrangian $L(\mu)$ when μ is substituted by w^{*}. This is because $\mathbf{x}^{*}$ is feasible for (LP). Furthermore, it satisfies (IV.23). Therefore $\mathbf{x}^{*}$ is an optimal solution for $L(w^{*})$. The Lagrangian Optimality Property implies that $L(w^{*})$ is the optimal objective function of (LP).

Lagrangian relaxation is a general solution strategy for solving mathematical programs that permits us to decompose problems to exploit their special structure. As such, this solution approach is perfectly tailored for solving many models with embedded network structure.

The reviewed Dantzig-Wolfe decomposition, Bender's decomposition and Lagrangian relaxation are methods that can be considered to work in two (or more) levels. The importance of this will be seen when the economic interpretation of one of them (Dantzig-Wolfe) is presented later in Chapter IX.

V AGGREGATION METHOD

A. INTRODUCTION

The idea underlying this approach is that of consolidating 'neighboring' origins (destinations). Balas (1965) uses aggregation to produce an algorithm of specific application to the solution of the transportation problem. In his approach, the solution to the aggregated problem is used as a means for obtaining some information that allows for a simplified version of the original problem. Progress can be made by considering at each step only a certain part of the problem's data. We will present this method, due to Balas, posed on a transportation problem.

The scheme consists of the following general pattern:

(i) Simplify (aggregate) the problem, reducing its size (this could be considered an auxiliary step for (ii)) ;

(ii) With the information obtained from (i), construct a derived problem, which is smaller than the original problem, although of the same order of magnitude, and solve it;

(iii) Hopefully, (ii) will solve the original problem. If not, some iterative process is applied that gradually increases the size of (ii). (An unfortunate sequence of iterations could end up reproducing the original problem, but it is not likely).

Figure 4 is a representation of the problem to be solved. In the figure, (a) is the original problem. The aggregation process groups 'contiguous' nodes to form a single aggregated node. This gives the schematic representation (b). (A numerical example is offered in Appendix B).

B. NOTATION AND DEFINITIONS

For notation purposes, consider the original set of supply nodes to be arranged in m groups. Groups are denoted as:

Group M_1: 1, 2, ..., m_1;

Group M_2: m_1+1, m_1+2, ..., m_2;

Group M_3; m_2+1, m_2+2, ..., m_3;

....

Group M_{m-1}; $m_{m-1}+1$, $m_{m-1}+2$, ..., m_m (V.1)

Similarly, destination nodes are arranged in 'n' groups as:

Group N_1: 1, 2, ..., n_1;

Group N_2: n_1+1, n_1+2, ..., n_2;

Group N_3; n_2+1, n_2+2, ..., n_3;

....

Group N_{n-1}; $n_{n-1}+1$, $n_{n-1}+2$, ..., n_n (V.2)

Let H be the set of all indices 'h' from (V.1), and K the set of all indices 'k' from (V.2). Then the original problem will be called hereafter ***Problem (I)***, and is stated in the following terms:

Problem (I)

$$\min \sum_{h\epsilon H} \sum_{k\epsilon K} c_{hk}x_{hk} \quad \text{(V.3a)}$$

$$\text{s.t.} \sum_{k\epsilon K} x_{hk} = a_h \quad \text{(V.3b)}$$

$$\sum_{h\epsilon H} x_{hk} = b_k \quad \text{(V.3c)}$$

$$x_{hk} \geq 0 \quad \text{(V.3d)}$$

Next we define the aggregated problem as ***Problem (II)*** as follows:

(i) Origin nodes belonging to group M_i will form the aggregated node i, with I being the set of all i;

(ii) Destination nodes belonging to group N_j will be aggregated as node j, with J being the set of all j.

The costs, supplies and demands for the aggregated problem (II) are defined as follows:

(iii) $C_{ij} = \min \{c_{hk} \mid h \in H, k \in K\}$ (V.4a)

$A_i = \sum_h a_h \quad$ with $h \in M_i$ (V.4b)

$B_j = \sum_k a_k \quad$ with $k \in N_j$ (V.4c)

$|C_{ij} - c_{hk}| \leq \alpha$, for all $h \in M_i$, $k \in N_j$, $i \in I$, $j \in J$, and some α (V.4d)

Then, the aggregated problem (II) is stated as

Problem (II)

$$\min \sum_{i \in I} \sum_{j \in J} C_{ij} X_{ij} \quad \text{(V.5a)}$$

$$\text{s.t.} \sum_{j \in J} X_{ij} = A_i \quad \text{(V.5b)}$$

$$\sum_{i \in I} X_{ij} = B_j \quad \text{(V.5c)}$$

$$X_{ij} \geq 0 \quad \text{(V.5d)}$$

Figure 4 is a representation of both problems.

Let $\bar{\mathbf{x}}$ be a nondegenerate feasible solution of (II). Associated with Problem (II) consider a new problem, called the *partial problem* related to $\bar{\mathbf{X}}$. For simplicity, hereafter

we will refer to this problem as ***Problem (III)***, implicitly assuming it is related to a flow coming from (II). The problem is formed according to the following

RULE: Consider only those flows in the original Problem (I) going from groups M_i to groups N_j such that $\overline{X}_{ij} > 0$.

Better than stating Problem (III), the reader is encouraged simply to follow the example in Appendix B.

Given the arrangement of Problem (I) it is easy to see that every component X_{ij} of the flow X has an associated *block* of flows in Problem (I). Those are the flows going from M_i to N_j. So, a total of mxn blocks of flow form the whole Problem (II). The above rule reduces the size of the problem to be solved, by excluding all those blocks of flows associated with zero flow in the solution's components to the Problem (II). This is particularly interesting in long transportation problems, where a lot of flows are at level zero in the optimal solution.

Clearly, we can establish a 1-1 mapping between feasible and nondegenerate solution to (II) and problems (III). (This is because a solution of such class for (II) is defined by the flow values corresponding to 'm + n - 1' nonzero variables).

Furthermore, for every feasible solution to problem (II) there exists a feasible solution to problem (III). It suffices to consider the flow

$$\overline{x}_{hk} = \left(\frac{a_h b_k}{A_i B_j} \right) \overline{X}_{ij} .$$

Also any feasible solution to a Problem (III) is a feasible solution to the corresponding Problem (I). (Problem (III) is more restricted than (I)).

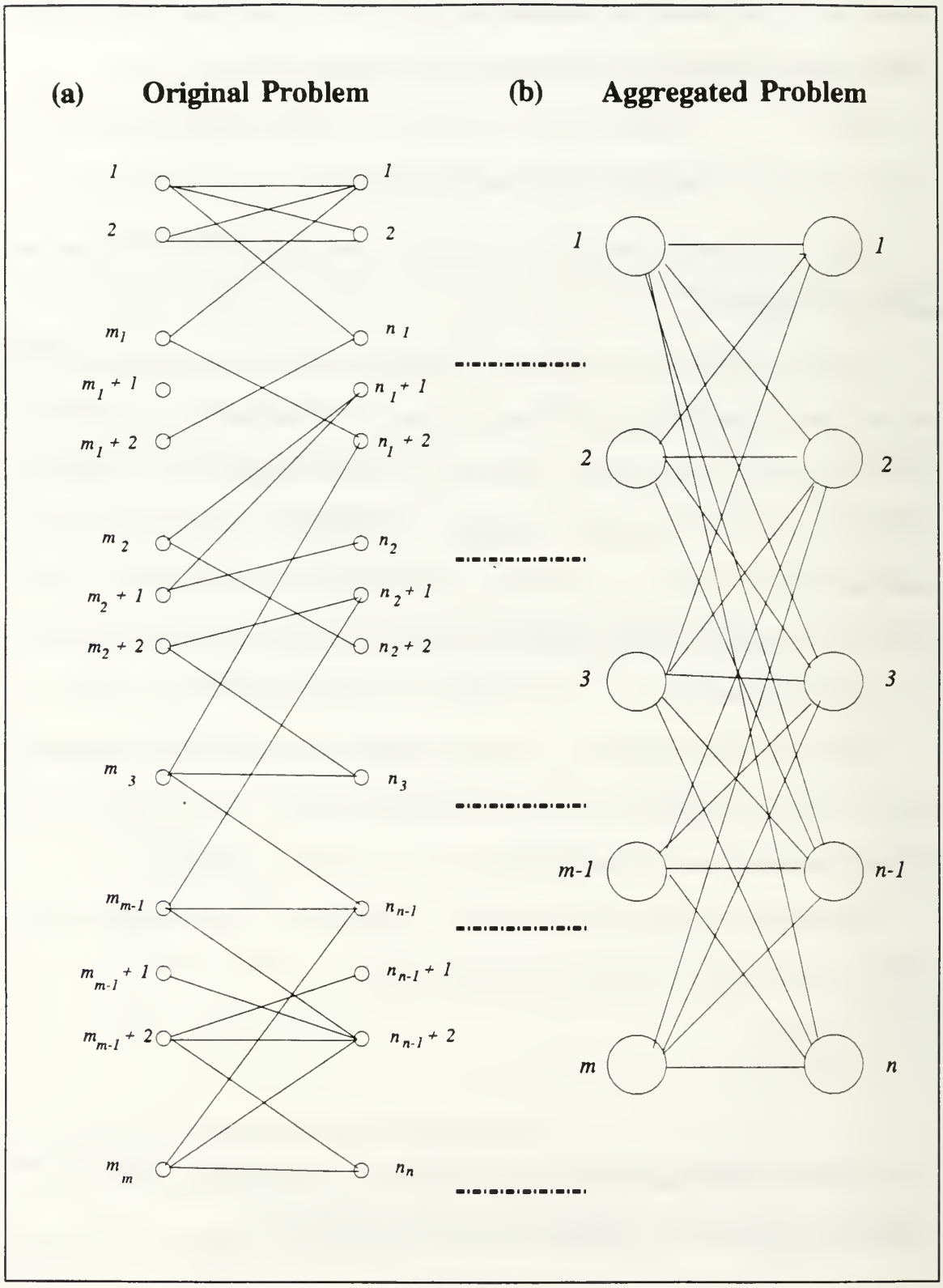

Figure 4.- *Balas' Node Aggregation.*

C. OPTIMALITY: RELATIONSHIP BETWEEN PARTIAL AND ORIGINAL PROBLEMS.

Here we use the complementary slackness, expressed in terms of the slack variables (Chapter II). The resulting condition is:

$$D_{hk} = c_{hk} - u_h - v_k \geq 0, \quad \text{for valid values of h,k} \qquad (V.6)$$

where D_{hk} represents a slack variable, and u_h , v_k are duals. The expression (V.6) will be zero for positive flows, and greater than zero for flows equal to zero.

Let D'_{hk} be the values of the slack variables in those blocks not in the optimal solution to (II). Then

Theorem (I): An optimal solution to problem (III) is also optimal for problem (I) if and only if $D'_{hk} \geq 0$.

Proof: Let $\bar{\mathbf{x}}'$ be an optimal solution to (III). Then it is feasible solution for (I). And since it satisfies the complementary slackness conditions it is also an optimal solution for (I).

Now suppose $\bar{\mathbf{x}}$ is an optimal solution for problem (I) and not for problem (III). Then there will be some positive flow $\bar{x}_{hk}$ not in the optimal solution for (III). The corresponding slack variable D'_{hk} will be less than zero, which contradicts the assumption in the theorem.

Next is an algorithm suggested by Balas (1965).

Algorithm:

(1) From Problem (I) formulate Problem (II);

(2) Solve Problem (II);

(3) From the optimal solution to (II) formulate Problem (III);

(4) Solve Problem (III);

(5) Check if, at this point, the solution satisfies Theorem (I).

(i) if solution to (III) is optimal for (I) stop;

(ii) if not, form a new Problem (III) adding those blocks in (III) violating Theorem (I).

(6) Improve the solution obtained in (4) so as to make it optimal for this new partial problem; Go to (5).

In Appendix B a numerical example develops in practical terms the preceding algorithm.

Balas' aggregation is a two-level method to be applied only to the solution of transportation problems. In the next chapter, more general multilevel methods are presented to solve more general network flow problems.

VI SCALING

A. INTRODUCTION

Scaling techniques give the best worst-case running time for many of the network optimization problems. They are computationally effective in cases where reoptimization from a good starting solution is more efficient than solving the problem from scratch. Gabow (1984) demonstrates that under a "similarity assumption" (that the logarithm of the largest magnitude in the data is of the order of the number of nodes) scaling could be used to advantage in designing theoretically efficient algorithms for a broad variety of network optimization problems.

Scaling was introduced by Edmonds and Karp (1972). Using scaling, the first polynomial time algorithm for the minimum cost problem was produced. The algorithm of Edmonds and Karp uses scaling on the right hand side repeatedly to employ the out-of-kilter method iteratively. Between 1972 and 1984, there was little interest in employing the scaling approach of Edmonds and Karp in actual network flow computation. In spite of favorable asymptotic worst-case performance it was widely presumed that algorithms employing data scaling would not be nearly as fast in practice as the network simplex method. Since 1985, research has been extensive. Researchers now recognize that scaling techniques have great theoretical value as well as potential practical significance (see, for example, Ahuja, *et al* , 1993).

B. NOTATION AND DEFINITIONS

Let G = (N, A) be a (directed) network defined by a set N of 'n' nodes and a set A of 'm' directed arcs. The following notation holds:

d(i): the demand of node i;

c_{ij} : the cost associated with the arc (i,j);

C : the largest cost;

u_{ij} : the capacity of the arc (i,j);

U : the largest capacity;

r_{ij} : the residual capacity of the arc (i,j).

A ***pseudoflow*** $\bar{x}$ is a function x: **A** -> **R** satisfying only the capacity and nonnegativity constraints (it need not satisfy the mass balance constraints).

The ***imbalance*** of node i is given by

$$e(i) = d(i) + \sum_{\{j:\ (j,i) \in A\}} x_{ji} - \sum_{\{j:\ (i,j) \in A\}} x_{ij}$$

all $i \in A$ (VI.1)

e(i) is called ***excess*** (***deficit***) if it is greater (lesser) than 0.

The set of nodes with excess (deficit) is denoted E (or D).

Next we define the ε-***optimality*** of a pseudoflow $\bar{x}$. This concept is, independently, due to Bertsekas (1979) and Tardos (1985). Figure 5 is a *kilter diagram* that graphically compares ε-*optimality* and the usual optimality.

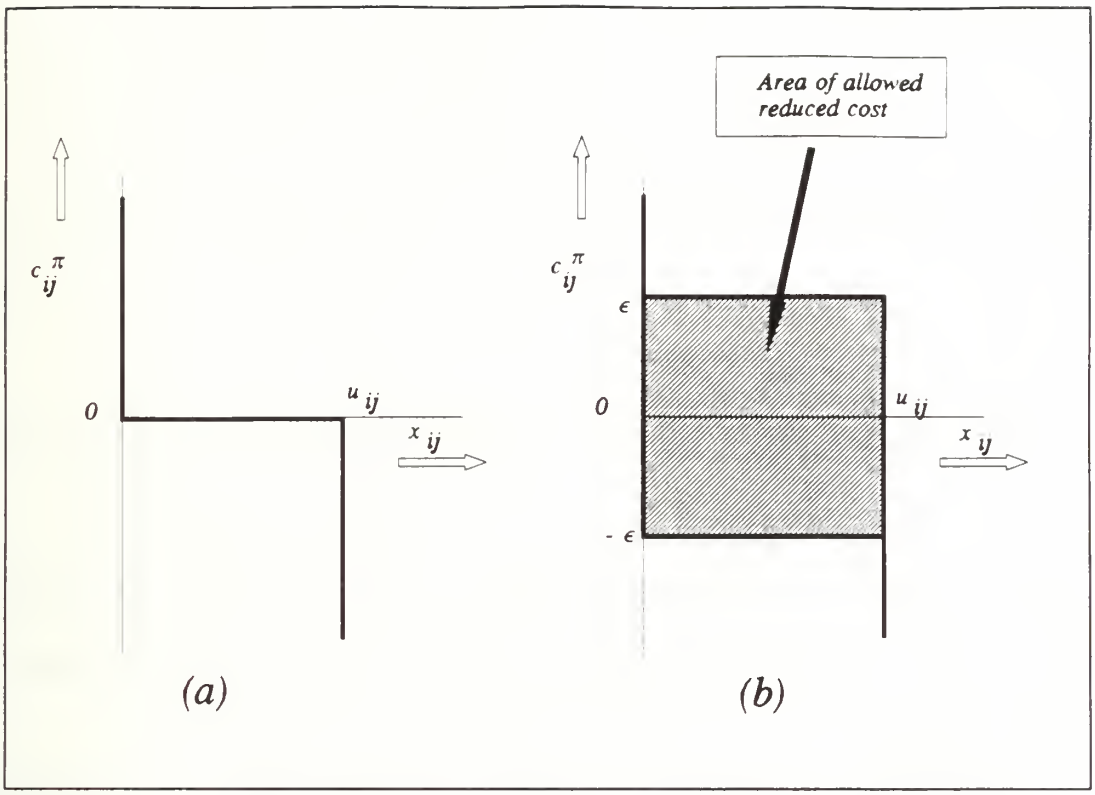

Figure 5.- *Kilter Diagrams showing: (a) Optimality; (b) ε-Optimality.*

A pseudoflow $\bar{x}$ is said to be ε-***optimal*** (for some $\varepsilon > 0$) if for some set of node potentials π the pair $(\bar{x}, \pi)$ satisfies:

$$c_{ij}^{\pi} > \epsilon \Rightarrow \bar{x}_{ij} = 0 \qquad \text{(VI.2a)}$$

$$-\epsilon \leq c_{ij}^{\pi} \leq \epsilon \Rightarrow 0 \leq \bar{x}_{ij} \leq u_{ij} \qquad \text{(VI.2b)}$$

$$c_{ij}^{\pi} < -\epsilon \Rightarrow \bar{x}_{ij} = u_{ij} \qquad \text{(VI.2c)}$$

In terms of the residual network, the pseudoflow $\bar{x}$ is said to be ε-optimal (for some $\varepsilon > 0$) if for every arc (i,j) in the residual network $G(\bar{x})$:

$$c_{ij}^{\pi} \geq -\epsilon \qquad \text{(VI.3)}$$

C. COST SCALING

As its name suggests, cost scaling is an algorithm approach that applies scaling to the costs. Assume, without great loss of generality, that the costs are integer. Then:

Theorem VI-1 (Bertsekas, 1986): For a minimum cost flow problem with integer costs, any feasible flow is ε-*optimal* whenever $\varepsilon \geq C$ (largest cost). Moreover, if $\varepsilon < 1/n$ then any ε-*optimal* feasible flow is an optimal flow.

Proof: Let $\bar{\mathbf{x}}$ be any feasible flow and let $\pi = 0$. Consider the residual network $G(\bar{\mathbf{x}})$. Then for every arc (i,j) in $G(\bar{\mathbf{x}})$ $c_{ij}^{\pi} = c_{ij} \geq -C$, therefore $\bar{\mathbf{x}}$ is ε-*optimal* for $\varepsilon = C$.

Now consider an ε-*optimal* flow $\mathbf{x}^*$ with $\varepsilon < 1/n$. Let π be the node potentials associated to that ε-*optimal* flow. Let W be any directed cycle in the residual network $G(\mathbf{x}^*)$. Then, by (VI.3)

$$\sum_{\{(i,j)\in W\}} c_{ij}^{\pi} \geq -\epsilon n > -1$$

But the costs are integers, and this implies that the above sum is nonnegative. Now, by Corollary II-2,

$$\sum_{\{(i,j)\in W\}} c_{ij} \geq 0$$

Therefore W is not a negative cost cycle. Hence, by Theorem II-1 $\mathbf{x}^*$ must be optimal.

A node i is said to be an ***active node*** when its imbalance is positive. An arc (i,j) adjacent from an active node, and belonging to the residual network, is said to be an ***admissible arc*** if $-\frac{1}{2}\epsilon \leq c_{ij}^{\pi} < 0$. In Figure 6 for an arbitrary arc (i,j), (a) represents a situation of optimality, while (b) illustrates admissibility.

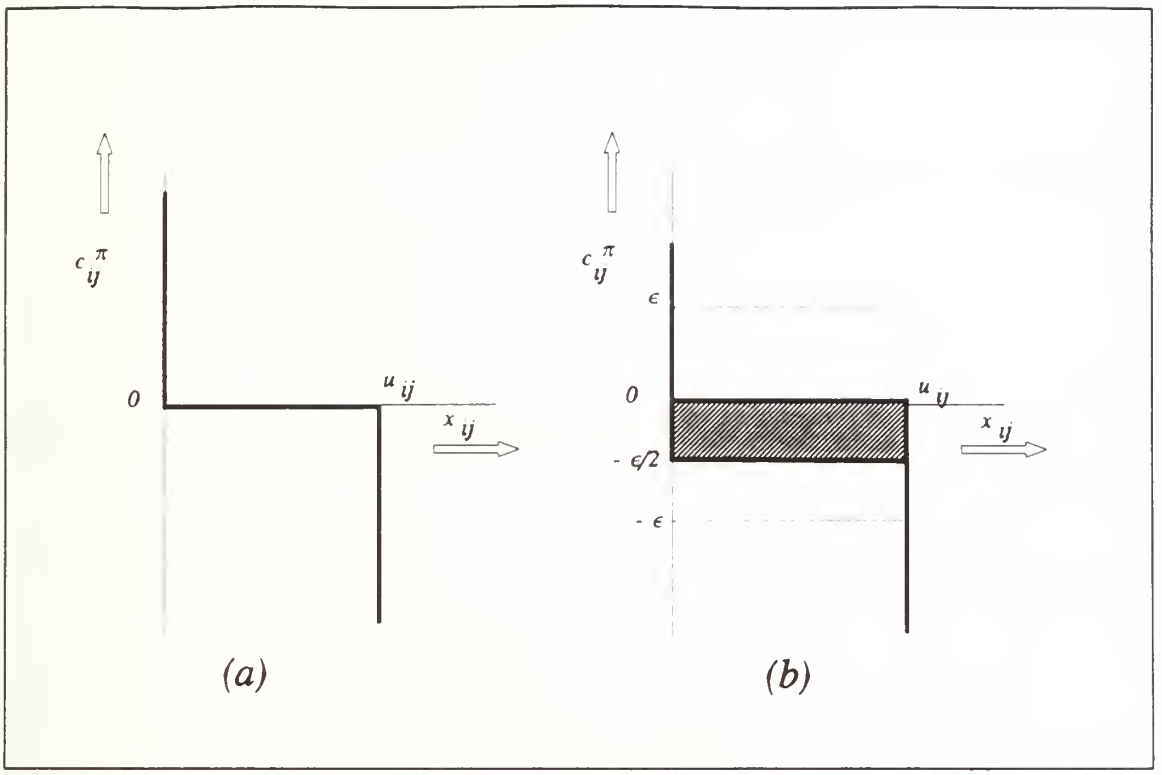

Figure 6.- *Kilter Diagrams showing: (a) Optimality; (b) Admissibility.*

Next we describe a generic-cost algorithm for minimum cost flows problems which works in two stages: first, it converts an ε-optimal flow into a (1/2 ε)-optimal pseudoflow; then gradually converts this pseudoflow into a flow, maintaining the (1/2 ε)-optimality. Figure 7 shows the algorithm, while Figures 8 and 9 are graphic representation of the successive steps on an example. Figure 8 starts with a situation determined by the first residual network, and ε = 8. The order of the operations performed has been determined to satisfy the example's purpose. Note that the values of the node potentials remain constant until Figure 9(d), when a situation occurs in which the active node has no associated admissible arc.

In the example, the demands of the nodes can be computed using formula (VI.1) as follows:

```
Algorithm COST SCALING

BEGIN

  π := 0;      ε := C;
  x is any feasible flow

  WHILE ε ≥ 1/N  DO BEGIN

      FOR every arc (i,j) in the network DO BEGIN
        IF          positive reduced cost then   x_ij  :=  0;
        ELSE IF negative reduced cost then   x_ij  :=  u_ij;
        ELSE      leave the flow on that arc unchanged;
      end; {for every ...}

      FOR all nodes in the network compute node imbalances e(i);

      WHILE the network contains an active node DO BEGIN
        Select an active node 'i';
        IF there is any admissible arc (i,j) in the Residual Network then begin
           push  δ = min {e(i), r_ij }   units of flow from 'i' to 'j';
           update imbalances;
        end;
       ELSE  π(i) := π(i) + ε/2
      end; {while the network ...}

    ε = ε/2

  end; {while ...}

END;
```

Figure 7.- ***Algorithm Cost Scaling.***

d(1) = e(1) - 10 + 0 = -10;

d(2) = e(2) - 35 + 10 = 5;

d(3) = e(3) - 0 + 30 = 20;

d(4) = e(4) - 0 + 5 = -15;

The imbalances for step (f) are calculated in the same fashion:

e(1) = d(1) + 0 - 0 = -10;

e(2) = d(2) + 0 - 0 = 5;

e(3) = d(3) + 0 - 20 = 0;

e(4) = d(4) + 20 - 0 = 5.

D. CAPACITY SCALING

Capacity scaling produced the first polynomial-time algorithm for the minimum cost problem (Edmonds and Karp, 1972). We present an algorithm that scales capacities due to Orlin (1988). It uses the concept of the ***Δ-Residual network G(x,Δ)***, defined to be the subgraph of the residual network G(x), induced by the arcs (forward or reverse) having a residual capacity of at least Δ units. For sake of clarity let us define also:

Δ-Excess Node: A node having an excess of at least Δ units;

Δ-Deficit Node: A node having a deficit of at least Δ units;

S(Δ): Set of Δ-Excess Nodes;

T(Δ): Set of Δ-Deficit Nodes.

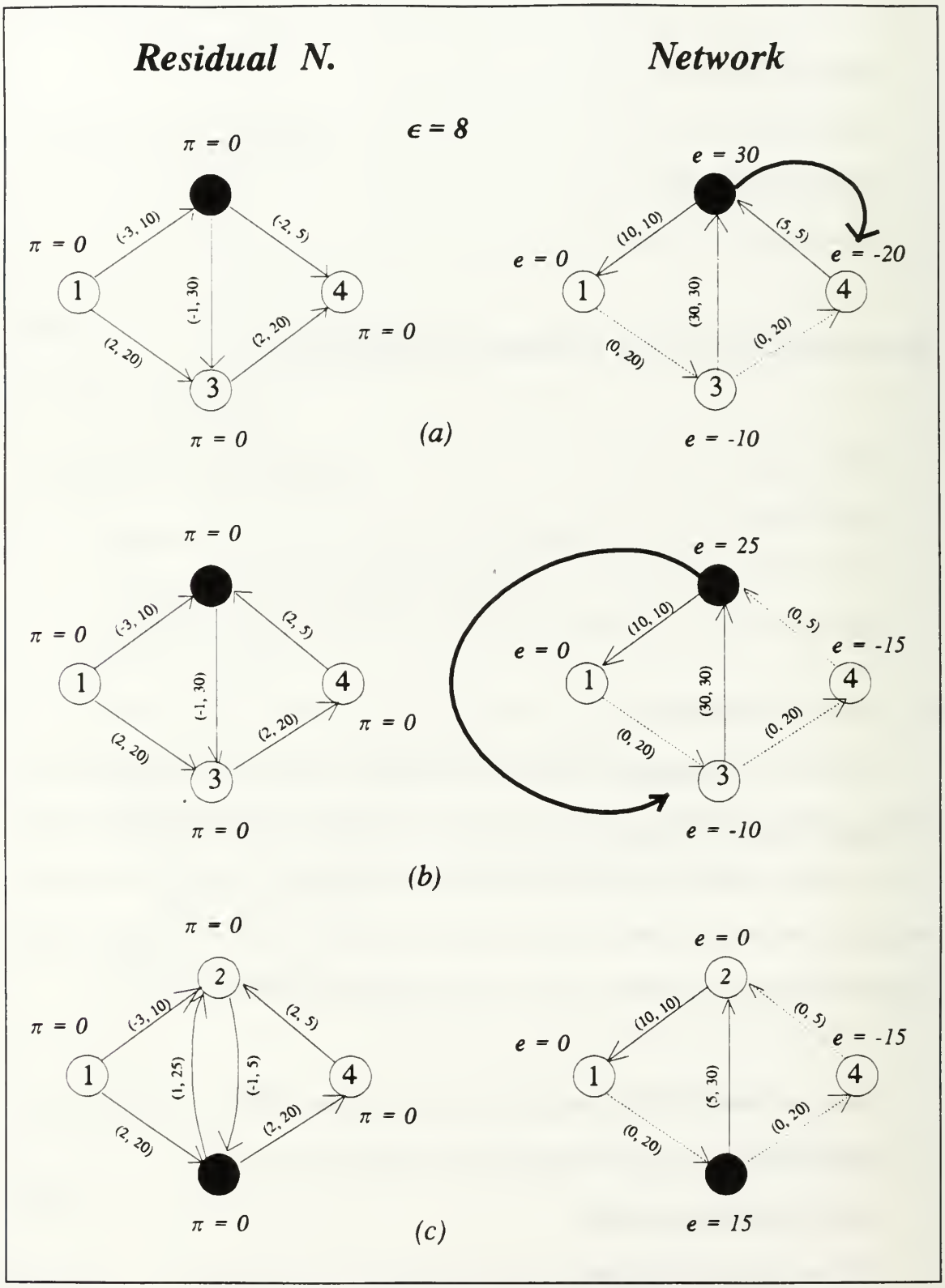

Figure 8.- *Generic Cost Scaling Algorithm. Example (a)*

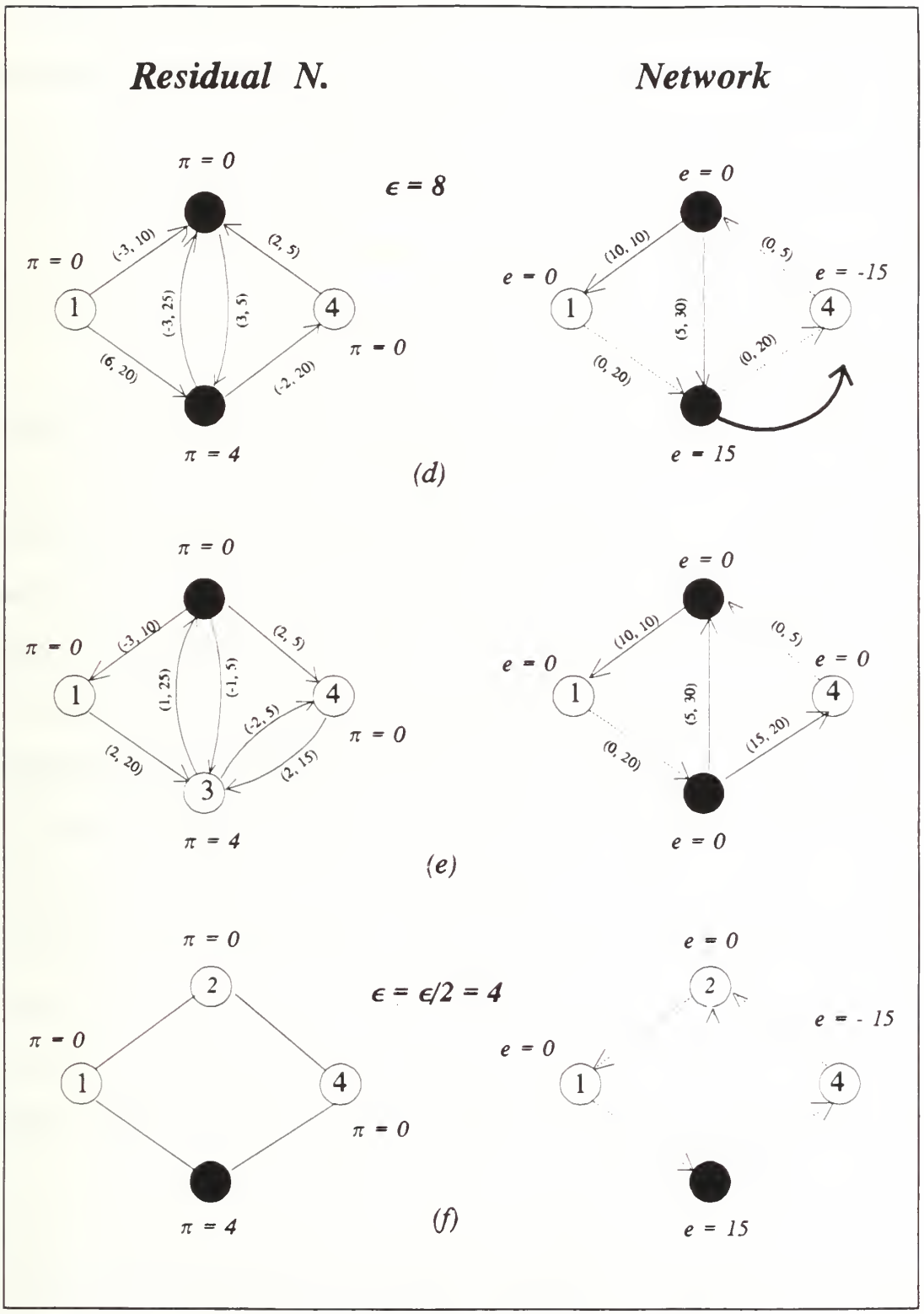

Figure 9.- *Generic Cost Scaling Algorithm. Example (b)*

The technique used is to send flow from a node of $S(\Delta)$ to a node of $T(\Delta)$. The flow is sent along a *direct* path defined in the Δ-residual network. Such a technique guarantees that every time a flow is sent, it will be sent in "packages" of Δ units.

The algorithm maintains nonnegative reduced costs in the Δ-residual network (*reduced cost optimality*). A pseudocode is showed in Figure 10.

In case that more than one path is available, we choose the most favorable (that with a minimum reduced cost).

Each scaling phase of the algorithm is identified by the value of the scale parameter Δ. The scale parameter is halved when each phase is finished. Within O(log U) scaling phases, $\Delta = 1$, and by the requirement that the data be integer, every node imbalance will be zero at the end of this phase. At this point $G(x, \Delta)$ is identical to $G(x)$. Since every arc in $G(x, \Delta)$ satisfies the reduced cost optimality conditions, the flow obtained is optimal. Figure 10 shows pseudocode for the algorithm.

Figures 11 through 14 describe the capacity scaling algorithm on a simple network representing a transportation problem with two supply nodes and two demand nodes. Two additional nodes (super-source, s, and super-sink, t) are included. The algorithm is represented step-by-step. Figure 11 is a description of the simplified data to appear on Figures 12 through 14. So, at every step three different pictures summarize the network status. On the left hand side the real network is shown, with data of flows, capacities and costs of each arc. Next to each node a number appears when there is an excess or deficit associated. The following describes the main features used in the graphics:

In the real network (left hand side) thick lines represent positive flow, while dotted lines mean zero flow. Next to the graph the current status of the sets $S(\Delta)$ and $T(\Delta)$ is also shown.

The two small networks represented on the right hand side are the residual (upper) and Δ-residual (lower), respectively. In those, continuous lines represent *direct* arcs, and dotted curved lines represent *reverse* arcs.

Thick lines in the Δ-residual network correspond to the next flow movement from a node another node. Two different circumstances motivate flow interchange:

(i) Existence of a path in the Δ-residual network connecting a node in $S(\Delta)$ with a node in $T(\Delta)$, as in Figures 12(a), 13(d-e) or 14(e);

(ii) Existence of an arc with negative reduced cost in the residual network, as in Figures 12(c) or 14(d). In such cases, the value of the reduced cost is shown in the center of the arc. Recall that those arcs will drive flow to saturation, so that only arcs with nonnegative reduced costs remain in the (residual) network.

Nodes with excess or deficit are filled in white, while all others are filled in black.

The following are the noteworthy situations in the execution of the example:

- **Figure 12(b):** no direct path is available connecting $S(\Delta)$ and $T(\Delta)$. Therefore the parameter scale must be halved.
- **Figures 12(c) and 14(d):** after halving the parameter scale, arcs with negative reduced costs presumably occur in the residual network. This forces flow to be sent along these arcs to bring them to saturation.
- **Figure 14(e):** the node 's' has been isolated in the residual network, with no imbalance. Thereafter, this node cannot be accessed in any Δ-residual network. The distance labels to this node will be infinite. This situation is consistent with the requirement of saturating all the arcs going to the super-sink.

Algorithm ***Capacity Scaling***

begin

$\pi := 0;$
$\Delta := 2^{\log U};$

WHILE $\Delta \geq 1$ ***DO BEGIN***

FOR *every arc (i,j) in G(x)* ***DO BEGIN***
IF $r_{ij} \geq \Delta$ *AND* $c^{\pi}_{ij} < 0$ *then* ***begin***
send r_{ij} *units of flow along arc (i, j); {saturate arc (i,j)}*
update x and the imbalances e
end;

$S(\Delta) = \{i \in G(x, \Delta)$, *such that* $e(i) \geq \Delta\}$
$T(\Delta) = \{i \in G(x, \Delta)$, *such that* $e(i) \leq -\Delta\}$

WHILE $(S(\Delta) \neq \Theta$ *AND* $T(\Delta) \neq \Theta)$
AND *(there exists a path from* $S(\Delta)$ *to* $T(\Delta))$ ***DO BEGIN***
select a node $k \in S(\Delta)$ *and a node* $l \in T(\Delta)$
determine shortest path distances d from node k to all other nodes in $G(x, \Delta)$ *with respect to the reduced costs* c^{π}_{ij}.
Select a node $q \in T(\Delta)$; *Let P(k, q) be the shortest path from k to q*
Augment Δ *units of flow along the path P(k, q)*
update $\pi = \pi - d$
update x, $S(\Delta)$, $T(\Delta)$, *and* $G(x, \Delta)$;
end; *{while}*

$\Delta = \Delta/2$

end *{while}*

end

Figure 10.- ***Algorithm Capacity Scaling***

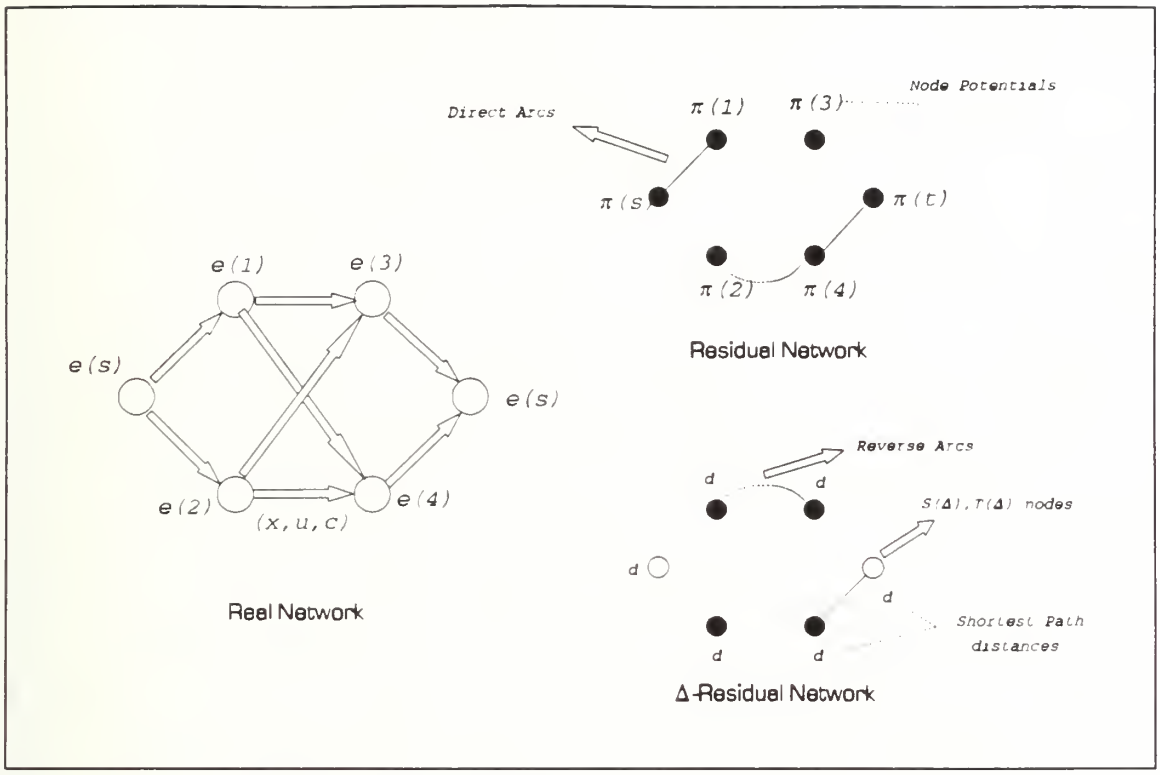

Figure 11.- ***Key for Flows in Figures 12, 13 and 14***

Scaling techniques maintain certain parallelism with multigrid, that will be proposed in the "Conclusions" section in Chapter IX. Also, it is remarkable that scaling suggests a more abstract concept of a grid, represented in the scaling parameter. But this concept requires a previous exposure to the concept of grids, which is deferred to Chapter VIII.

In the next chapter, we present a methodology that has been used in dynamic systems control problems. It is known as the Perturbation Method, and, although somewhat apart from the general path of this research, the fact that aggregation is a basic mechanism in the development of the perturbation method, together with the special treatment it gives to some linear programs is of special interest since it presents more insights into the process of building up a multigrid approach to optimization problems.

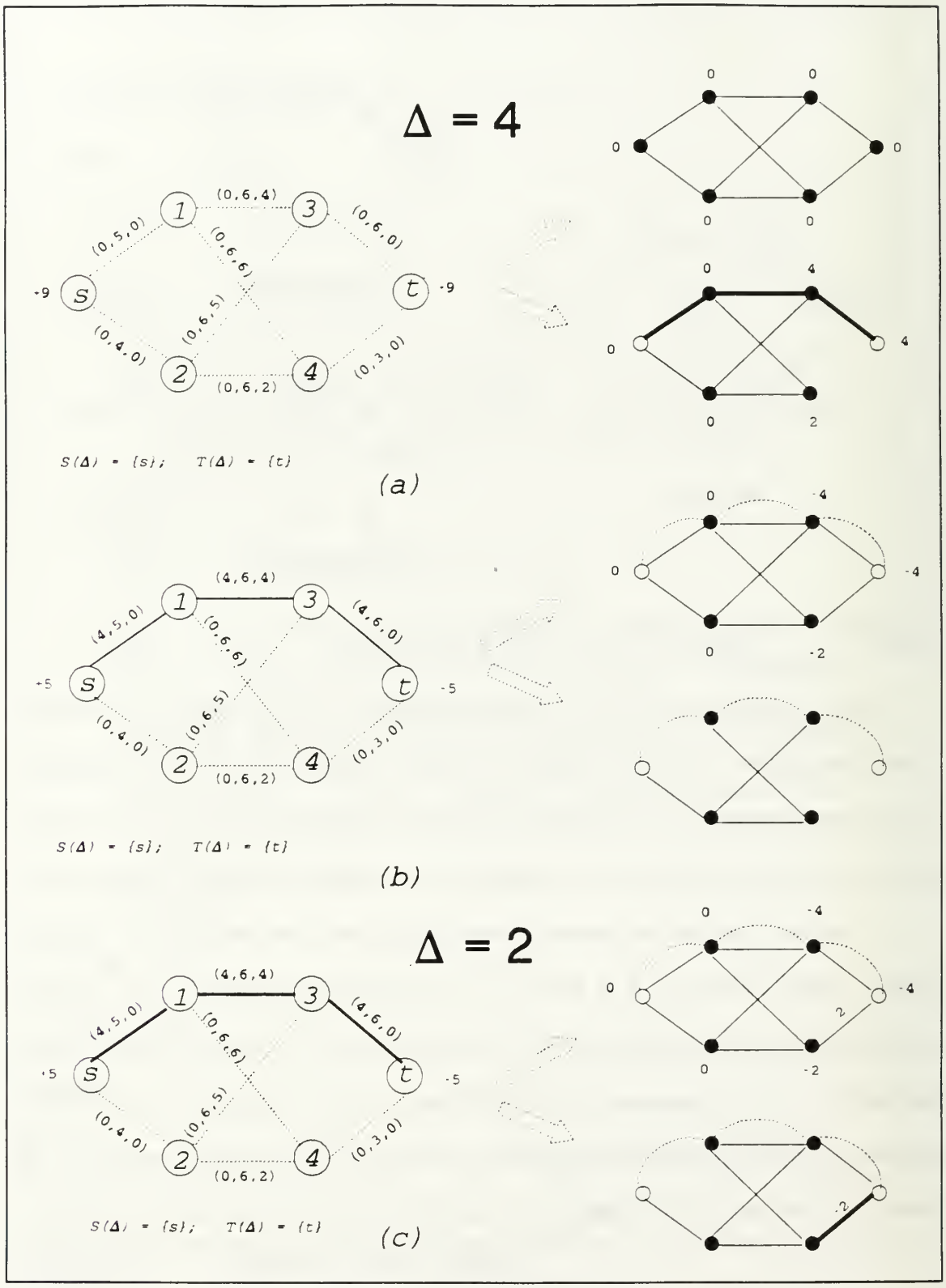

Figure 12.- ***Example: Capacity-Scaling Algorithm (I)***

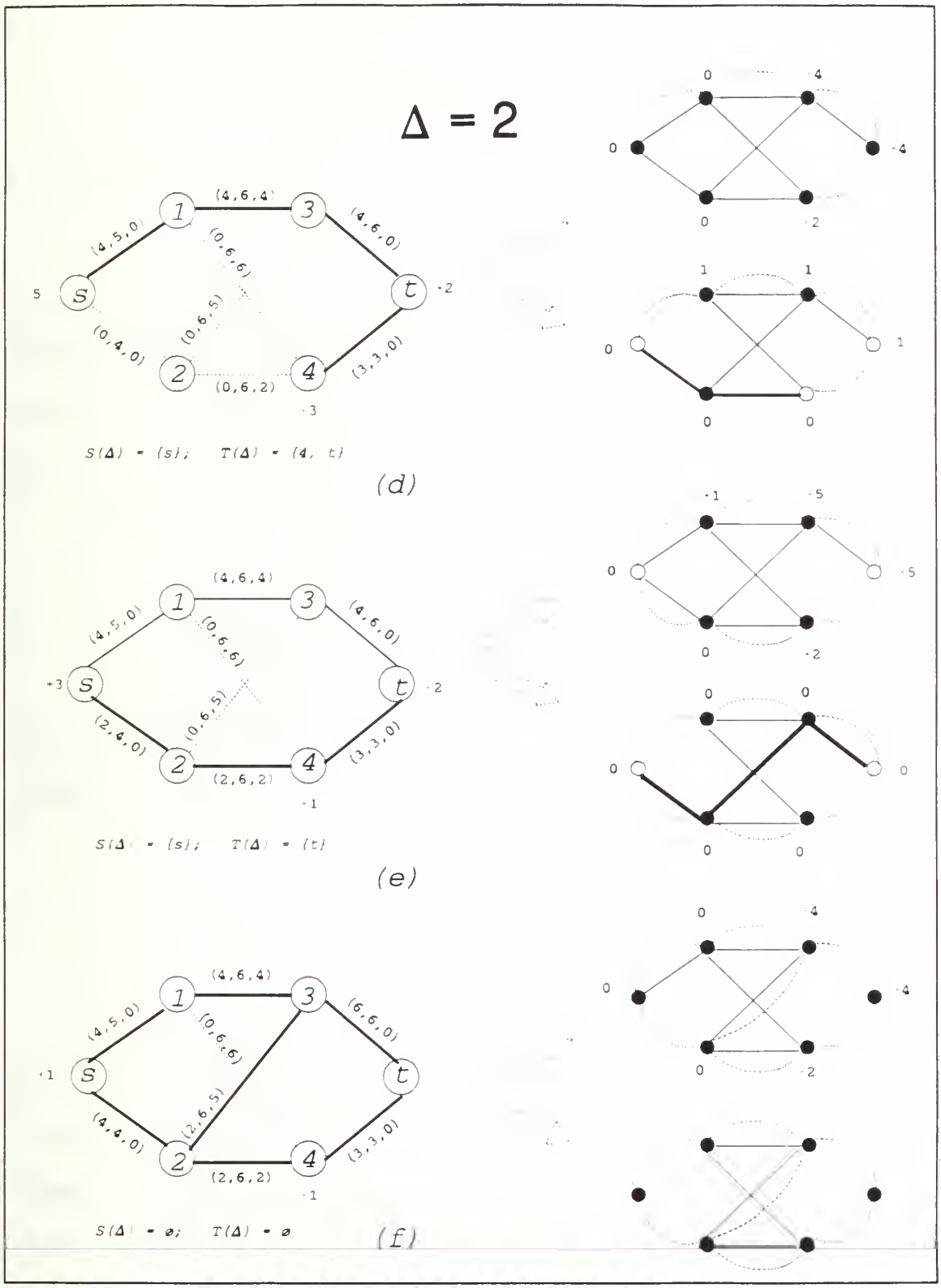

Figure 13.- ***Example: Capacity-Scaling Algorithm (II).***

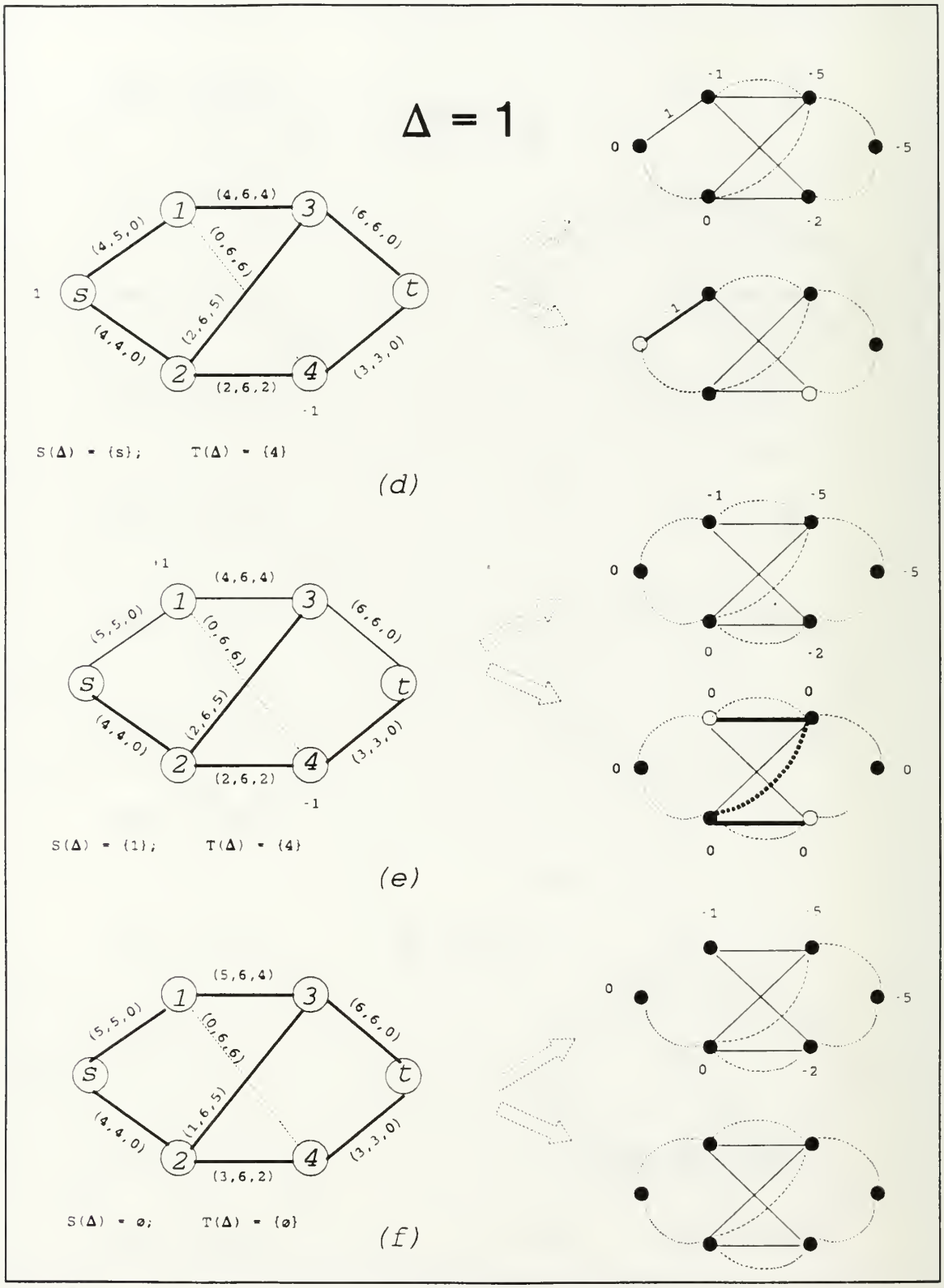

Figure 14.- ***Example: Capacity-Scaling Algorithm (III)***

VII PERTURBATION METHOD

A. PERTURBATION

The main idea of the perturbation method is to transform an original problem into a simpler one - called the "reduced problem" - . The original problem will then be considered a "perturbation" of the reduced problem. The perturbations are supposed to be small, and their magnitude is characterized by a parameter ε. We will consider only the case of linear programs.

Consider a mathematical program $F^{\cdot}(0)$:

$$\begin{aligned} \min \quad & c_0^T x \\ \text{s.t.} \quad & A_0 x \geq 0 \end{aligned} \tag{VII.1}$$

where $\mathbf{x}$ is an n-vector and $\mathbf{A}_0$ is an mxn matrix of constraints, and consider the program $F^{\cdot}(\varepsilon)$: .

$$\begin{aligned} \min \quad & c^T(\epsilon) x \\ \text{s.t.} \quad & A(\epsilon) x \geq 0 \end{aligned} \tag{VII.2}$$

The case of study is when we need to solve a problem type (VII.2) for a particular value of the parameter ε. Suppose that a direct solution to (VII.2) is difficult, but a solution to the "reduced program" (VII.1) is comparatively simple. Then it is reasonable to solve the original program by successive approximations, using a procedure in which the first phase is the solution to the reduced program, and the second phase is the computation of a correction to it.

B. SIMPLICITY AND PROXIMITY.

The success of the perturbation method relies on two factors: simplicity and proximity.

Linear programs are the simplest class of programs. Within this class, complexity increases with the problem size. But for any fixed size, obtaining a solution can be simplified if one uses the specific structure of the programs considered. Given that both the original and the reduced problem belong to the same class of problems (i.e. linear programs), the next factor to consider in order to determine simplicity will normally be the possibility of admitting direct aggregation or decomposition, discussed below.

A formal measure of proximity of the original to the reduced problem is

$$\Delta = \sup \left[\sup_{x \in \bar{X}} |c(x) - c_0(x)|, \quad \sup_{x \in \bar{X}} |A_{l.}(x) - A_{0l.}(x)|, \quad l = 1, \ldots, w \right] \qquad \text{(VII.3)}$$

Generally speaking, the smaller Δ is, the closer the solution to the original problem will be to that of the reduced problem. One would like to express the error of the solution approximation as a function of the proximity.

Some types of structure admitting direct simplification are considered next. They can be classified as:

(a) Directly decomposable programs;

(b) Directly aggregatable programs;

(c) Directly aggregatable and decomposable programs.

1. Directly Decomposable Programs

Let a program have the form

$$\min \quad c_1 z_1 + c_2 z_2 + \ldots + c_k z_k \tag{VII.4}$$

where $z_i = f_i(x^i)$, the different x^i being vectors of disjoint subspaces of the global space containing the vectors $\mathbf{x}$ satisfying the linear program (VII-4). And let the objective function be monotonically increasing (i.e. the c's positive). Then to solve this program it is sufficient to solve k independent subproblems

$$\begin{aligned} &\min \quad f_s(x^s) \\ &\text{s.t.} \quad x^s \in X_s, \qquad s = 1, 2, \ldots, k \end{aligned} \tag{VII.5}$$

A linear program is close to being directly decomposable when it has the form

$$\begin{aligned} &\min \quad \mathbf{c}\,\mathbf{z} + \varepsilon\, \mathbf{c}'\,\mathbf{x} \\ &\text{s.t.} \quad \mathbf{A}_0\,\mathbf{z} + \varepsilon\, \mathbf{A}_1\,\mathbf{x} \geq \mathbf{0} \end{aligned} \tag{VII.6}$$

where

$$\mathbf{x} = (x^1, x^2, \ldots, x^k)$$

$$\mathbf{z}_s = \mathbf{z}_s(x^s), \qquad x^s \in X_s$$

These are problems in which the state of each subsystem is determined by its "own" vector $\mathbf{x}^s$, and the activities of each subsystem depend weakly on those of other subsystems.

2. Directly Aggregatable Programs

Those are problems of the form

$$\begin{aligned} &\min \quad \mathbf{c}\,\mathbf{z} \\ &\text{s.t.} \quad \mathbf{A}\,\mathbf{z} \geq \mathbf{b} \end{aligned} \tag{VII.7}$$

where $\mathbf{z} = z(\mathbf{x})$, $\mathbf{x}$ is n-dimensional vector.

Suppose one can solve the aggregated problem

$$\begin{aligned} &\min \quad \mathbf{c}\,\mathbf{z} \\ &\text{s.t.} \quad g(\mathbf{z}) \geq 0 \end{aligned}$$

If $\mathbf{z}^*$ is a solution to the aggregated problem then $x^* = z^{-1}(z^*)$, when solvable, is a solution to the original problem.

EXAMPLE. The following simple transportation problem (represented in Figure 15):

$$\min\ 3\,(x_{11} + x_{12}) + 2\,(x_{13} + x_{14}) + 4\,(x_{21} + x_{22}) + 5\,(x_{23} + x_{24})$$

$$\begin{aligned} \text{s.t.} \quad & x_{11} + x_{12} + x_{13} + x_{14} = 10 \\ & x_{21} + x_{22} + x_{23} + x_{24} = 14 \\ & x_{11} + x_{21} = 8 \\ & x_{12} + x_{21} = 5 \\ & x_{13} + x_{23} = 4 \\ & x_{14} + x_{24} = 7 \\ & x_{ij} \geq 0 \end{aligned}$$

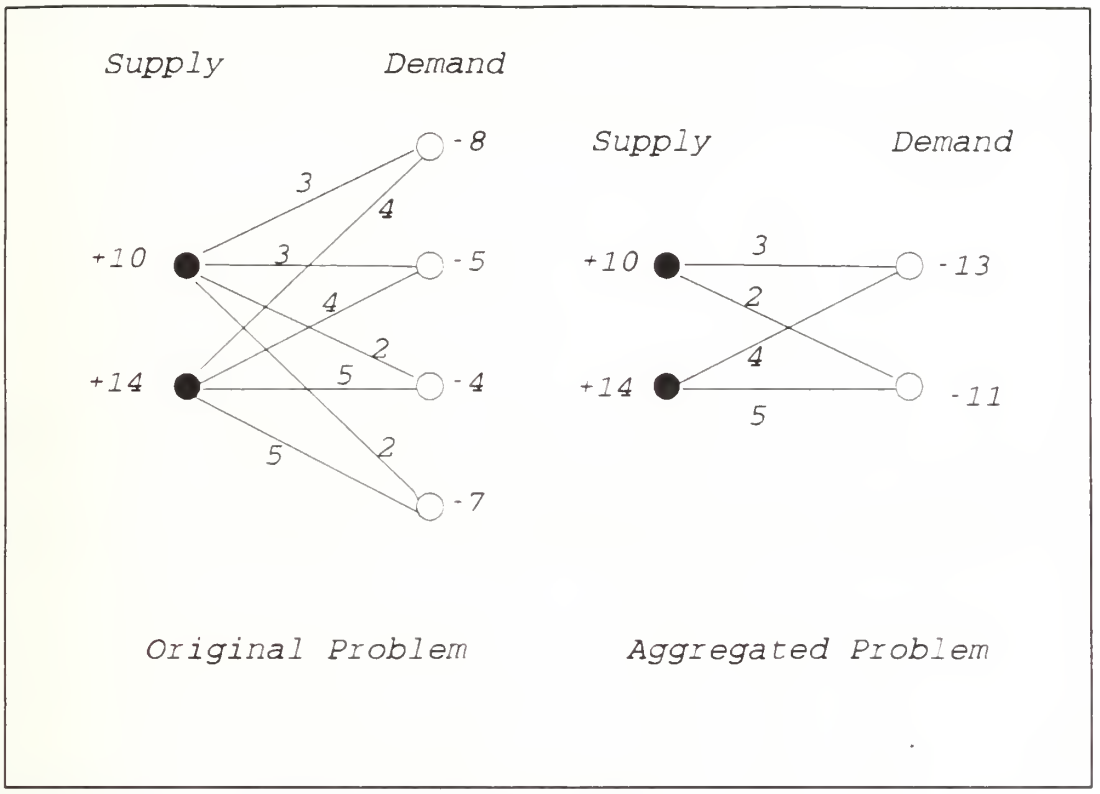

Figure 15.- *Example of Direct Aggregation*

can be directly aggregated by grouping the variables in the following form, suggested by the parentheses above:

$$z_{11} = x_{11} + x_{12}$$

$$z_{12} = x_{13} + x_{14}$$

$$z_{21} = x_{21} + x_{22}$$

$$z_{22} = x_{23} + x_{24}$$

This produces the "aggregated" problem,

$$\min\ 3\ z_{11} + 2\ z_{12} + 4\ z_{21} + 5\ z_{22}$$

s.t.

$$z_{11} + z_{12} = 10$$

$$z_{21} + z_{22} = 14$$

$$z_{11} + z_{21} = 13$$

$$z_{12} + x_{22} = 11$$

$$z_{ij} \geq 0$$

In this case, we aggregated pairs of demand nodes having the same transportation costs per unit. The solution to the aggregated problem is:

$$z_{11} = 0$$
$$z_{12} = 10$$
$$z_{21} = 13$$
$$z_{22} = 1$$

and the optimal value of the objective function is 77.

Now if we solve the system

$$x_{11} + x_{12} = 0$$
$$x_{13} + x_{14} = 10$$
$$x_{21} + x_{22} = 13$$
$$x_{23} + x_{24} = 1$$
$$x_{11} + x_{21} = 8$$
$$x_{12} + x_{22} = 5$$
$$x_{13} + x_{23} = 4$$
$$x_{14} + x_{24} = 7$$

we can obtain the following solution (not unique):

$$x_{11} = x_{12} = 0$$
$$x_{13} = 3; \quad x_{14} = 7;$$
$$x_{21} = 8; \quad x_{22} = 5;$$
$$x_{23} = 1; \quad x_{24} = 0;$$

which can be verified to solve the original transportation problem.

3. Directly Aggregatable and Decomposable Problems

Consider the problem

$$\begin{aligned} \min \quad & c_1 z_1 + c_2 z_2 + \ldots + c_k z_k \\ \text{s.t.} \quad & A(z_1, z_2, \ldots, z_k) \geq 0, \end{aligned} \tag{VII.8}$$

where $\mathbf{z}_i = z_i(\mathbf{x}^i)$, $\mathbf{x}^s \in X_s$, $s = 1,2,\ldots,k$. Here the objective function does not have to be monotonically increasing. One can obtain solutions z_s^* to the aggregated problem,

$$\begin{aligned} \min \quad & \mathbf{c}\,\mathbf{z} \\ \text{s.t.} \quad & g(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k) \geq 0 \end{aligned} \tag{VII.9}$$

If there exist solutions $\mathbf{x}^{s*}$ of the equations $z_s(x^s) = z_s^*$ then the vector $x^* = (x^{s*})$ is a solution to the general problem.

EXAMPLE: Depicted in Figure 16 is a transportation problem with this structure. For sake of simplicity we avoid data in this example. It consists of two "independent" transportation subproblems, each of them with directly aggregatable structure. This structure is granted by a relationship between costs parallel to that of the example in section B.2 (i.e. costs from a node to successive and disjoint pairs of adjacent nodes are equal).

To solve the problem on the left we solve to optimality each of the two independent subproblems. To solve each subproblem, we can take advantage of the special structure determined by the costs relations above.

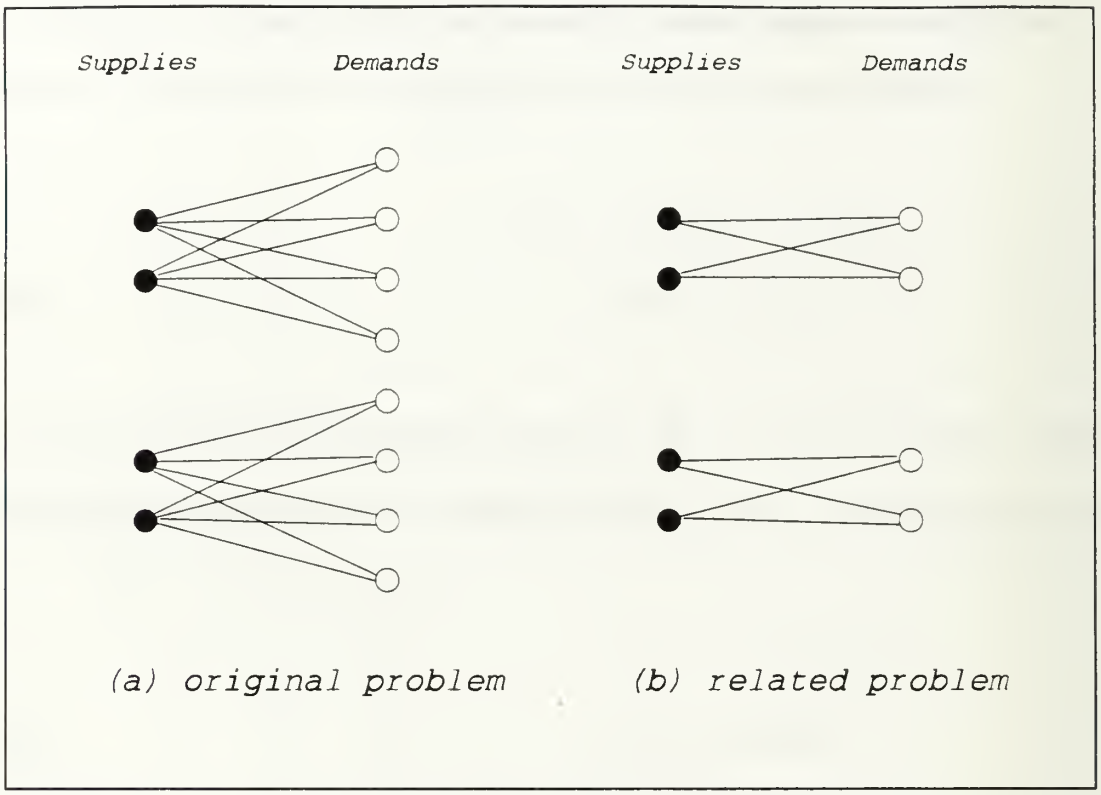

Figure 16.- ***Directly Aggregatable and Decomposable Problem.***

The details of such a problem are difficult and tedious, and it is felt that the underlying ideas may be more easily appreciated using the schematic presentation in Figure 16 than by exhaustive analysis.

C. OPTIMAL BASIC SET INVARIANCE

Let $F^{\cdot}(\epsilon)$ be the linear program defined as

$$\begin{aligned} \min \quad & c^{T}(\epsilon)\, x \\ \text{s.t.} \quad & A(\epsilon)\, x = b \\ & x \geq 0 \end{aligned} \qquad \text{(VII.10)}$$

with corresponding reduced problem

$$\begin{aligned} \min \quad & c_0^T x \\ \text{s.t.} \quad & A_0 x = b_0 \\ & x \geq 0 \end{aligned} \tag{VII.11}$$

Let $D^*(0)$ be the dual problem associated with $F^*(0)$, i.e.

$$\begin{aligned} \max \quad & b_0^T \lambda \\ \textit{s.t.} \quad & A_0^T \lambda \leq c_0 \end{aligned} \tag{VII.12}$$

Suppose that x_0^*, λ_0^* are optimal solutions to (VII.11) and (VII.12), respectively, and furthermore, they are unique. Then there exists a submatrix **B** of the matrix $\mathbf{A}_0$, such that

$$Bx_B^* = b_{0B} \tag{VII.13}$$

$$x_B^* > 0 \tag{VII.14}$$

$$B^T \lambda_0^* = c_B \tag{VII.15}$$

$$N^T \lambda_0^* > c_N \tag{VII.16}$$

where

$\mathbf{x}_B^*$ is an m-vector of basic variables, $x_B^* = (x_j^*, \quad j \in J_0)$

J_0 is the optimal basic index set (used next),

B is $\{A_j, \quad j \in J_0\}$ (submatrix formed by columns associated with the optimal basic index set),

N is $\{A_j, \quad j \notin J_0\}$

$\mathbf{c}_B$ is $\{c_{0j}, \quad j \in J_0\}$

c_N is $\{c_{0j},\ j \notin J_0\}$

The condition (VII.16) guarantees the uniqueness of the solution; (VII.14) is the nondegeneracy condition and guarantees the uniqueness of the solution λ_0^* of the dual problem.

The optimality conditions for the perturbed program are:

$$A(\epsilon)x(\epsilon) = b(\epsilon) \tag{VII.17}$$

$$x(\epsilon) \geq 0 \tag{VII.18}$$

$$A^T(\epsilon)\lambda(\epsilon) \geq c(\epsilon) \tag{VII.19}$$

$$x^T(\epsilon)[A^T(\epsilon)\lambda(\epsilon) - c(\epsilon)] = 0 \tag{VII.20}$$

here, (VII.17) and (VII.18) express the primal and dual feasibility; (VII.20) is the complementary slackness condition.

THEOREM (Optimal Basic Set Invariance): Let $A(\varepsilon)$, $b(\varepsilon)$, $c(\varepsilon)$ be continuous functions for $\varepsilon \in [0, \varepsilon']$. Let the solution to the reduced problem (2) be unique and nondegenerate. Then there exists a positive upper bound $\varepsilon'' \leq \varepsilon'$ such that, if $0 \leq \varepsilon \leq \varepsilon''$ then the optimal basic set of the perturbed program (1) is invariant, and its unique solution is given by

$$x^*(\epsilon) = [x_B^*(\epsilon) \quad x_N^*(\epsilon)]^T \tag{VII.21}$$

where

$$x_B^*(\epsilon) \quad \text{is} \quad \{x_j^*(\epsilon),\ j \in J_0\} = B^{-1}(\epsilon)b(\epsilon) \tag{VII.22}$$

$$x_N^*(\epsilon) = 0 \tag{VII.23}$$

$$\lambda^*(\epsilon) = (B^{-1})^T(\epsilon)c_B(\epsilon) \qquad \text{(VII.24)}$$

Note that the set J_0 consists of the indices in the base of the reduced problem. So the claim here is that the set of indices for both optimal solutions of the reduced and perturbed problems are the same. Then, to obtain a solution to the perturbed program it suffices to solve the reduced problem, determine the optimal basis index set, and solve (VII.13) to obtain $x_B(\epsilon)$. Finally, make $x_N(\varepsilon)$ and $x^*(\epsilon) = [x_B(\epsilon) \quad x_N(\epsilon)]$.

Sketch of proof: Let $x_0^* = [x_{0B}^* \quad x_{0N}^*]$ be the unique and nondegenerate optimal solution of the reduced problem. It must satisfy the optimality conditions (VII.13-16). Substituting (VII.13) into (VII.14) and (VII.15) into (VII.16),

$$B^{-1}(0)\ b(0) > 0 \qquad \text{(VII.25)}$$

$$B^T(0)\ [B^{-1}]^T(0)\ c_B(0) > c_N(0) \qquad \text{(VII.26)}$$

Now, if $\mathbf{A}(\varepsilon)$, $\mathbf{b}(\varepsilon)$, $\mathbf{c}(\varepsilon)$ are smooth enough, for ε sufficiently small

$$B^{-1}(\epsilon)\ b(\epsilon) > 0 \qquad \text{(VII.27)}$$

$$B^T(\epsilon)\ [B^{-1}]^T(\epsilon)\ c_B(\epsilon) > c_N(\epsilon) \qquad \text{(VII.28)}$$

Inequalities (VII.27-28) check the validity of $x^*(\epsilon)$ defined as in (VII.21) as an optimal solution to the perturbed problem.

So far, we are determining only the indices of the solution $\mathbf{x}(\varepsilon)$ that form the basis, but not the magnitudes of the associated components. This is achieved using the following corollary.

COROLLARY: Let $\mathbf{A}(\varepsilon)$, $\mathbf{b}(\varepsilon)$ be differentiable up to and including order m. Then there exists an ε'' such that, for all $\varepsilon \in [0, \varepsilon'']$, the solution $x_0^*(\epsilon)$ may be written in the form

$$x^*(\epsilon) = x_0^* + \epsilon x^{(1)} + \epsilon^2 x^{(2)} + \ldots + \epsilon^m x^{(m)} + o(\epsilon^{m+1}) \qquad \text{(VII.29)}$$

If

$$A(\varepsilon) = A_0 + \varepsilon A1 \qquad \text{(VII.30)}$$

$$b(\varepsilon) = b_0 + \varepsilon b_1 \qquad \text{(VII.31)}$$

$$c(\varepsilon) = c_0 + \varepsilon c_1 \qquad \text{(VII.32)}$$

then using conditions (VII.21-22) we can rewrite formula (VII.8) in the form

$$(A_0 + \epsilon A_1)\,(x_0^* + \epsilon x^{(1)} + \epsilon^2 x^{(2)} + \ldots + \epsilon^m x^{(m)} + o(\epsilon^{m+1})) = b_0 + \epsilon b_1$$

or, expressed in powers of ε:

$$(A_0 x_0^* - b_0) + \epsilon\,(A_1 x_0^* + A_0 x^{(1)} - b_1) + \epsilon^2\,(A_1 x^{(1)} + A_0 x^{(2)}) + \ldots = 0$$

in which the first term is identically zero, and the other terms give the following recursive relationships between successive vectors $\mathbf{x}^{(i)}$:

$$A_0 x^{(1)} = b_1 - A_1 x_0^*$$

$$A_0 x^{(2)} = - A_1 x^{(1)}$$

.....

$$A_0 x^{(k+1)} = - A_1 x^{(k)} \qquad \text{(VII.33)}$$

and using condition (VII.32)

$$F^*(\epsilon) = F_0^* + \epsilon \left[c_1^T x_0^* + (\lambda^*)^T (b_1 - A_1 x_0^*) \right] + o(\epsilon^2)$$

$$F^*(\epsilon) = F_0^* + \epsilon F_1^* + o(\epsilon^2), \quad 0 \le \epsilon \le \epsilon'' \qquad \text{(VII.34)}$$

Thus, we have a complete result for the case of uniqueness and nondegeneracy.

The results above can be generalized for the case of nonuniqueness, provided special conditions are met. Let conditions (VII.30-32) hold, and suppose further that

Θ^* is the set of solutions to the reduced problem,

Λ^* is the unique dual solution.

Then

$$F^*(\epsilon) = F_0^* + \epsilon F_1^* + o(\epsilon^2) \qquad \text{(VII.35)}$$

with

$$F_1^* = J_1^* + b_1^T \lambda^* \qquad \text{(VII.36)}$$

where J_1^* is the optimal value of the objective function for the following "auxiliary problem":

$$\min \; c_1 - A_1^T \lambda^*$$

$$x \in \Theta^* \qquad \text{(VII.37)}$$

The mentioned special condition requires the solution to the "auxiliary problem" must be unique and be also a nondegenerate solution to the reduced problem.

The application of the results above to linear programming problems can be summarized in the following outline of an algorithm:

(1) Solve the reduced program and its dual;

(2) If the solutions in (1) are unique, then J_0 determines the optimal basis for the perturbed problem;

(3) Determine $x(\varepsilon)$ as a solution to $B(\epsilon)x = b(\epsilon)$;

(4) If the solution to the reduced program is not unique, but it is nondegenerate, then

- determine the optimal set Θ^*
- Solve the "auxiliary problem" (VII.37);

(5) If the solution obtained in (VII.37) is unique, it determines the set J_0 and the matrix $B(\varepsilon)$. Then go to (3).

(6) If the solution obtained in (VII.27) is unique but degenerate then we need to determine the optimal set Λ of the dual program and to solve the dual auxiliary program;

(7) If the solution obtained in (6) is unique then the corresponding basis gives J_0 and $B(\varepsilon)$. Then go to (3).

Figure 17 represents this flow in a schematic manner.

The work presented up to this point is intended to familiarize the reader with some of the basic ideas of optimization, network problems and optimality conditions, and the optimization methods that could be considered as members of a wide family of multilevel methods (in which also some aspects of the techniques described in this Chapter can be included). We are in position to describe the peculiarities of the multigrid methodology, to be done next in Chapter VIII. Finally, the objectives of the research will be addressed in Chapter IX.

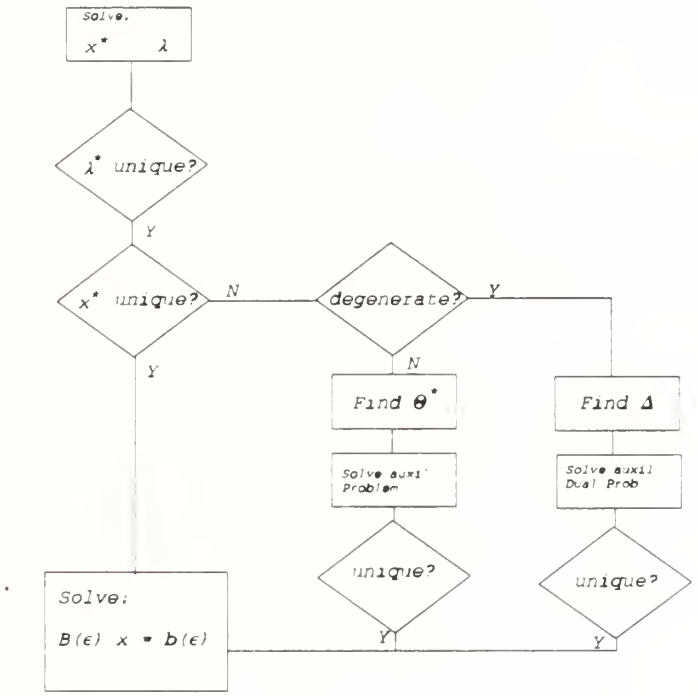

Figure 17.- *Perturbation Method. Flow Diagram.*

VIII. THE MULTIGRID METHOD

A. INTRODUCTION

Detailed descriptions of the multigrid method are given in Briggs, 1987, or Hackbusch, 1980. Here we will summarize the essential facts that could help in building the appropriate approach to solving certain types of optimization problems by means of that technique.

Consider the problem of heat propagation along a finite rod of unit length. This is a boundary value problem expressed by the second-order ordinary differential equation

$$-u'' + \sigma u = f, \quad 0 < x < 1, \quad \sigma \geq 0,$$
$$u(0) = u(1) = 0 \qquad \text{(VIII.1)}$$

where u and f are functions of x. Numerically, the problem can be solved using a ***finite difference*** method (Gerald, *et al, 1989).* The domain of the problem is partitioned into N subintervals of constant length $h = 1/N$. The partition so obtained defines a grid of $N+1$ points, including the end points 0 and 1. The resulting second-order finite difference approximation produces a system of linear equations

$$\mathbf{A}\,\mathbf{x} = \mathbf{f} \qquad \text{(VIII.2)}$$

The system (VIII.2) has the following special structure:

$$\frac{1}{h^2}\begin{bmatrix} 2+\sigma h^2 & -1 & & & & \\ -1 & 2+\sigma h^2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & -1 \\ & & & & -1 & 2+\sigma h^2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ \vdots \\ v_{N-2} \\ v_{N-1} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_{N-2} \\ f_{N-1} \end{bmatrix} \qquad \text{(VIII.3)}$$

Equation (VIII.3) is sometimes abbreviated by the ***difference stencil*** (Hackbusch, 1980), as follows:

$$h^{-2}\ [-1 \quad 2+\sigma h^2 \quad -1]\ v \quad = \quad f$$

Conventional iterative methods of solution use an initial guess $\mathbf{v}^{(0)}$ as initial solution. The *proximity* to the true solution **u** is measured by the ***error*** $\mathbf{e} = \mathbf{u} - \mathbf{v}^{(0)}$, which can be regarded as the *exact correction* to the (in this case) initial guess. The error is unknown. One way to estimate this proximity is by using the ***residual***. So, if **v** is an approximation to the true solution, the residual **r** is defined as:

$$\mathbf{r} = \mathbf{f} - \mathbf{A}\mathbf{v} = \mathbf{A}\mathbf{u} - \mathbf{A}\mathbf{v} = \mathbf{A}(\mathbf{u} - \mathbf{v}) = \mathbf{A}\mathbf{e} \qquad \text{(VIII.4)}$$

In (VIII.4) the residual **r** measures how well **v** solves (VIII.2).

The expression

$$\mathbf{A}\mathbf{e} = \mathbf{r} \qquad \text{(VIII.5)}$$

is called the ***residual equation***. It is important to note that the residual equation uses the same matrix as the original equation. This is the ***first key fact***.

Conventional iteration processes are represented as

$$\mathbf{v}^{(1)} = \mathbf{P}\,\mathbf{v}^{(0)} + \mathbf{g}, \tag{VIII.6}$$

where **P** is the iteration matrix. The error after n iterations is

$$\mathbf{e}^{(n)} = \mathbf{u} - \mathbf{v}^{(n)} = (\mathbf{P}\,\mathbf{u} + \mathbf{g}) - (\mathbf{P}\,\mathbf{v}^{(n-1)} + \mathbf{g})$$

$$= \mathbf{P}\,\mathbf{u} - \mathbf{P}\,\mathbf{v}^{(n-1)} = \mathbf{P}\,(\mathbf{u} - \mathbf{v}^{(n-1)}) = \mathbf{P}\,\mathbf{e}^{(n-1)},$$

therefore

$$\| \mathbf{e}^{(n)} \| = \| \mathbf{P} \|^{n} \| \mathbf{e}^{(0)} \|. \tag{VIII.7}$$

It can be shown that

$$\lim_{n \to \infty} \mathbf{P}^{n} = 0 \quad \text{if and only if} \quad \max_{i} | \lambda_i(\mathbf{P}) | < 1 \tag{VIII.8}$$

where $\lambda_i(\mathbf{P})$ are the eigenvalues of **P**. This suggests that <u>not all</u> linear systems of equations can be solved using these conventional iterative techniques. The success using those approaches will depend in the structure of the matrix **A** defining the problem.

Assuming the problem is solvable and the matrix **A** nonsingular, the error $\mathbf{e}^{(0)}$ corresponding to the initial guess of the solution can be expressed as

$$\mathbf{e}^{(0)} = \sum_{k=1}^{N-1} c_k \mathbf{w}_k \tag{VIII.9}$$

where c_k is a real coefficient, and w_k, k = 1,2,...,N-1 are the eigenvectors of **A**.

A ***second key fact*** is that the error is smoothed by conventional iterative methods. (See Chapter I). The fact allows that, after k iterations, $\mathbf{e}^{(k)}$ can be usually approximated with fewer points than $\mathbf{v}^{(k)}$. This can be exploited to represent $\mathbf{e}^{(k)}$ on a coarser grid.

In the case (VIII.1), the eigenfunctions have the form of sine functions (Fourier modes). The k^{th} eigenfunction is expressed as

$$w_k = C_k \sin (k \pi x) \qquad \text{(VIII.10)}$$

k is called the ***wavenumber***, equal to the number of half cycles which constitute **v** in the domain of the problem.

Correspondingly, the k^{th} eigenvector of the matrix **A** in (VIII.2), expressed as a one-dimensional array (vector) of N equi-spaced sample points has the following form for its components:

$$w_{k,j} = \sin\left(\frac{jk\pi}{N}\right) \qquad \text{with} \quad 1 \le k \le N\text{-}1 \quad \text{and} \quad 0 \le j \le N \qquad \text{(VIII.11)}$$

and we say that the vector is represented in a N-grid.

In multigrid, grids are denoted in terms of the distance 'h' between two consecutive points of the grid. If the mesh size is h then we denote the corresponding grid as Ω^h. For this last to make sense, it is necessary to have fixed the total length of the grid domain.

As an example of a class of problems in which multigrid techniques are successfully applied, Figures 18 and 19 represent the results of applying a conventional Jacobi iterative process to the system of linear equations corresponding to the heat propagation along a rod of unit length. The problem, stated above, is described as an example in Briggs, 1987.

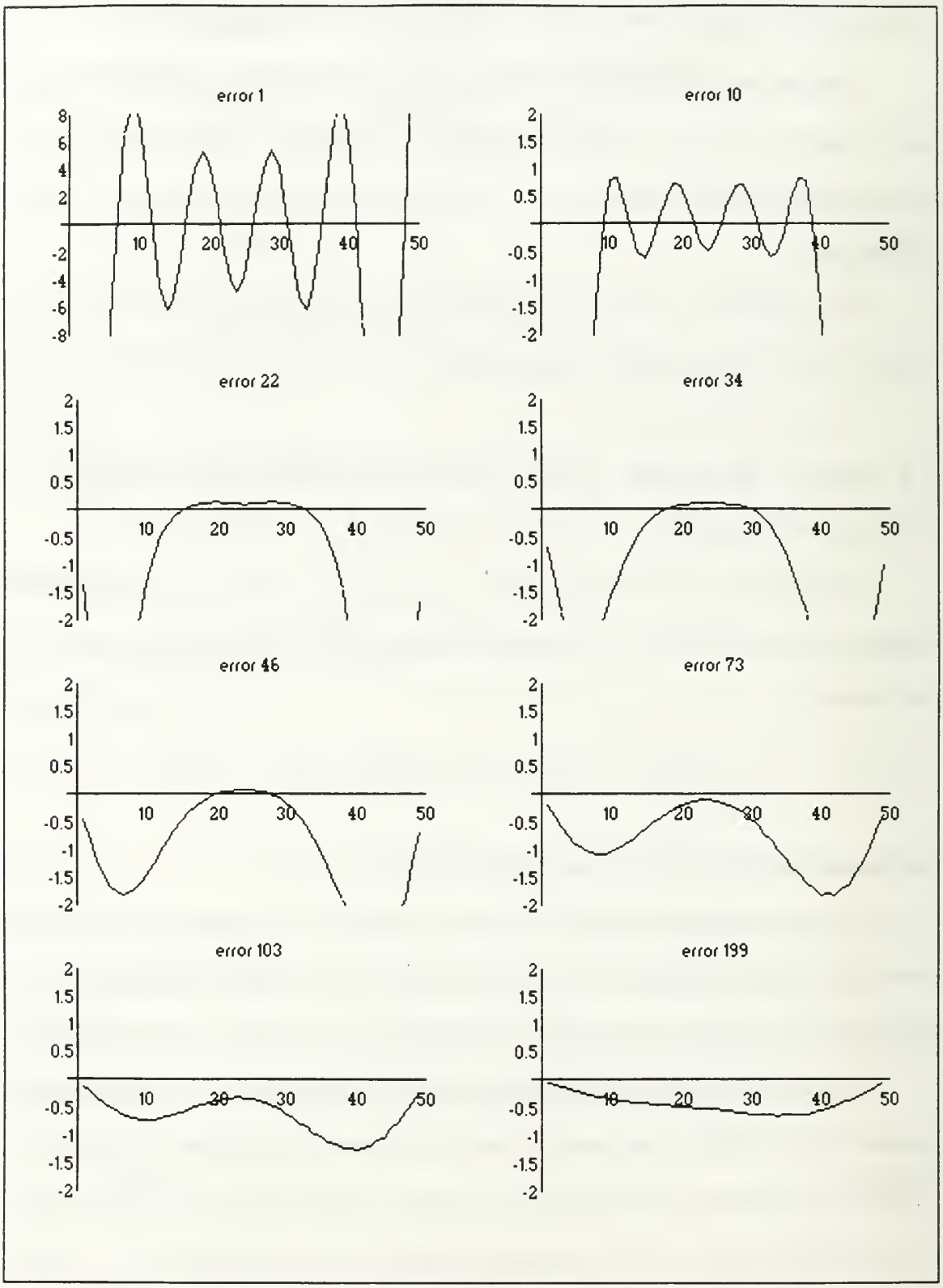

Figure 18.- *Change in the Error Vector.*

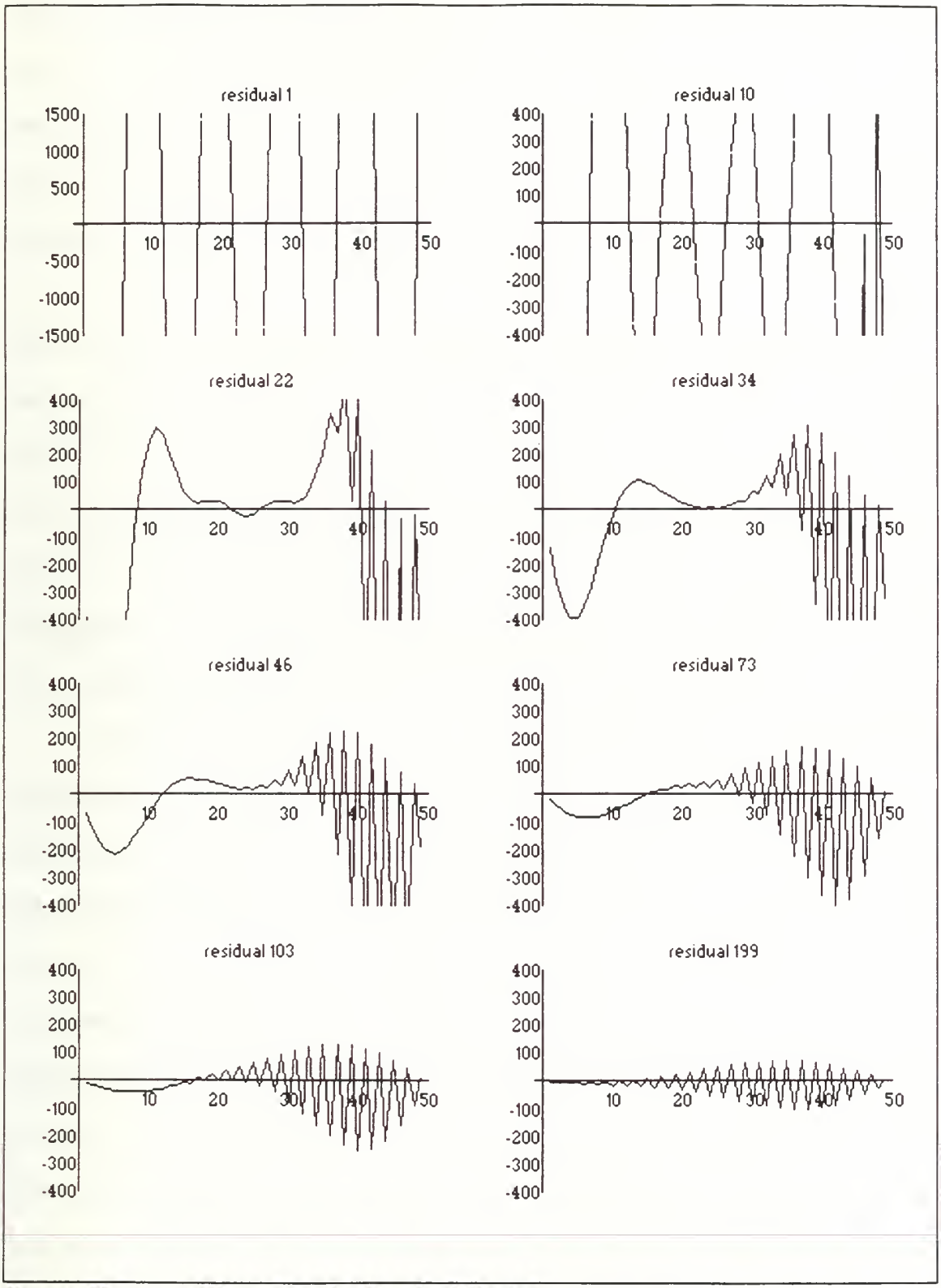

Figure 19.- *Change in the Residual Vector.*

Here we have modified the setting to be more general in that we allow the right hand side of the system to be other than zero. We chose the values of the right hand side to be integer random numbers. The details and the "Mathematica" code used to solve the problem are included as Appendix C. It also contains the numerical values of the right hand side vector, and some intermediate results that could be used for checking purposes.

A total of 200 Jacobi iterations were performed. Figure 18 shows the change in the error vector. An arbitrary oscillatory error was introduced as initial value, a consequence of the oscillatory initial guess for the solution. It can be seen that the error becomes less and less oscillatory, as the number of iterations increases. (It is easily seen that the number of crossings of the error curves through the x-axis decreases with increasing iterations). After 199 iterations, only a very smooth wave remains. This kind of behavior in the error is required for multigrid techniques to be successful. For completeness, we also graphed the change in the residual vector. This is represented in Figure 19.

A second example was constructed having the solution shown in Figure 20. This is a more oscillatory solution. The next figure shows the evolution of both error vector and current solution after a few iterations are performed. Notice the smoothing of the error, despite the lack of smoothness in the current solution.

The program included in Appendix C allows for an "animated" picture of both error and residual vectors for the first example. Also, it can be used for studying different values of the right hand side vector.

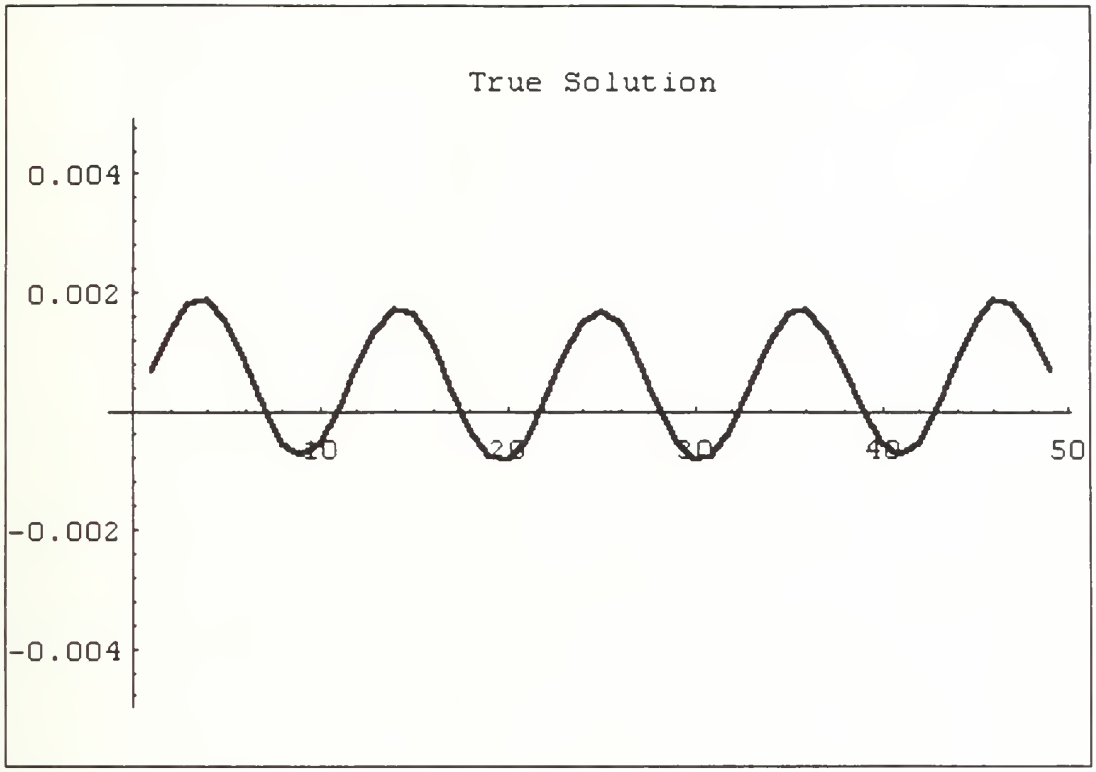

Figure 20.- *True Solution for Second Example.*

B. GENERAL MULTIGRID SCHEME.

In section A we have seen one condition for multigrid to work, i.e., the error vector should be smoothed as the iteration progresses. Now let us examine what happens to the error when appropriately "moved" from a finer grid to a coarser grid. Suppose that our error vector is sufficiently approximated by a wave with wavenumber k = 4 (Figure 22(a)), posed on a grid with 12 grid points, which we will refer to as Ω^h . Furthermore, suppose that there is no higher frequency component in the error vector (i.e. an iterative process has taken place that eliminated them previously). The former grid is capable of "detecting" frequency components with wavenumber less than k = 12, as can be seen in Figure 22(b)(c). Note that the wave with k = 12, N = 12 is indistinguishable from the wave having k=0, N=12. Higher k's cause an "***aliasing***" phenomena, depicted in Figure 23(d) (they are

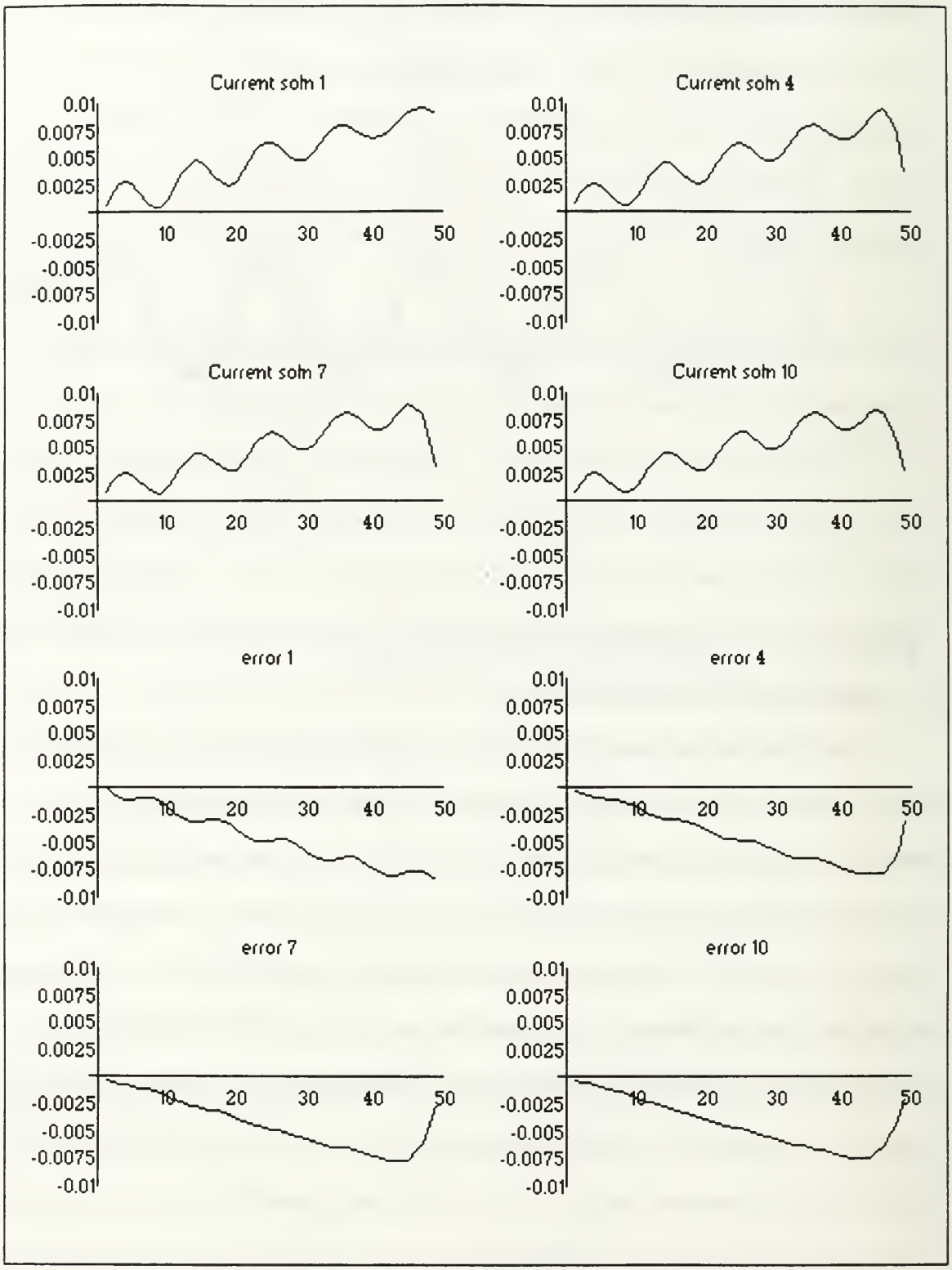

Figure 21.- *Second Example. Evolution of the Error Vector and Current Solution.*

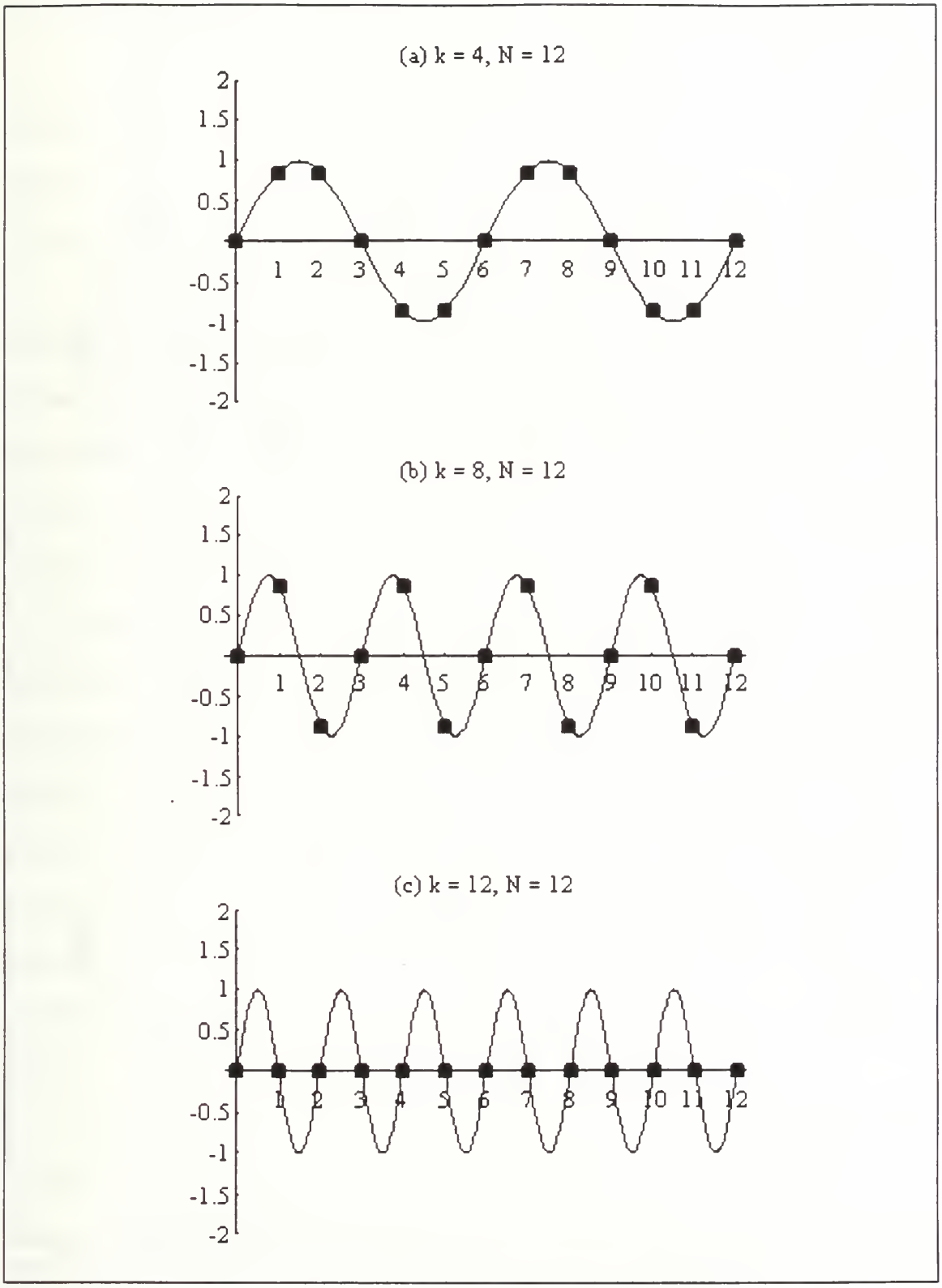

Figure 22.- *Representation of various components of the Error Vector.*

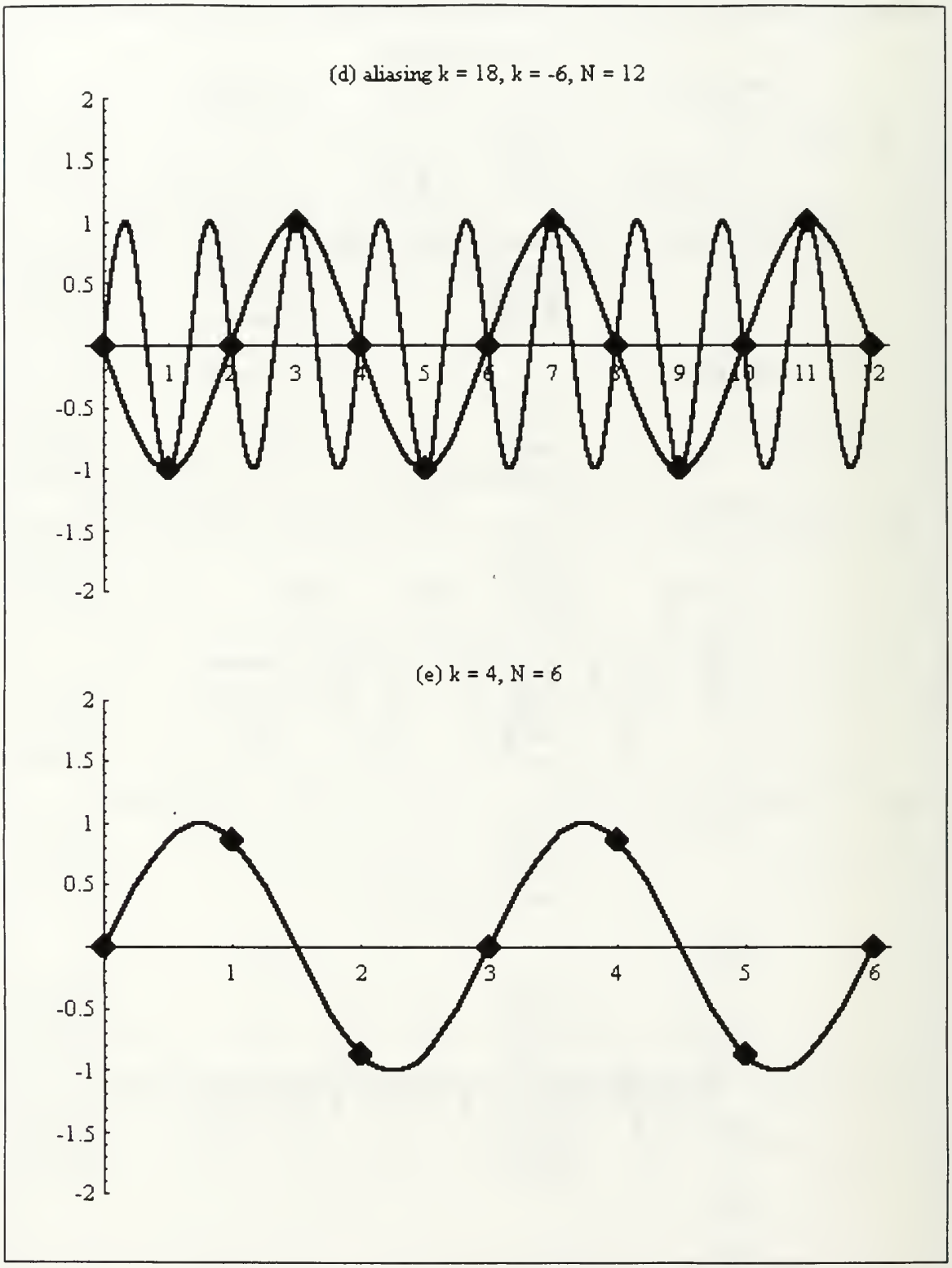

Figure 23.- *Representation of various spectrum components of the Error Vector (cont).*

confounded with other less oscillatory waves).

Let us go back to our error k = 4 wave. In the spectral representation of the "detectable" waves on the N = 12 grid (Figure 24(a)) this wave lies in the first third of the spectrum. We can say that it belongs to the class of the spectrum's ***low frequency components***, defined as those waves located in the first half of the spectrum.

Now let us express the corresponding spectrum for a N = 6 grid. This is done in Figure 24(b). Now the range of "detectable" waves goes up to k = 6. In this representation, the wave with k = 4 is located on the second/last third of the spectrum. We can say that this wave is a ***high frequency (or oscillatory) component***, a term defined in a similar fashion as in the preceding paragraph. Thus, our error vector becomes *more oscillatory* when transferred to the coarser grid. This is a ***third key fact***.

Figure 24 helps in understanding one of the basic mechanisms of the multigrid technique. Whenever the error vector is formed only by low frequency components in the relative context of a grid, we can force it to become more oscillatory by transferring it to another coarser grid. Typically, this is done doubling the grid mesh (i.e. halving the number of grid-points). This action is supported by invoking the ***fourth key fact*** : an error that is smooth on a grid Ω^h can be accurately transferred to a coarser grid Ω^{2h} and furthermore, the error representation in Ω^{2h} contains enough information to be interpolated to the next finer grid Ω^h and produce an accurate representation of the error there.

Now we can point out what the multigrid approach is in the above problem. Figure 18 represents the smoothing of the error vector when iteration is in progress. So we relax (iterate) on a grid until the error representation is "sufficiently smooth" on that grid. The

error then can be accurately represented on a coarser grid. This transfer is accomplished without loss of information. But now, the error representation on the coarser grid appears to be more oscillatory. A second relaxation process in this new grid will eliminate the "new" oscillatory components faster. When the relaxation process shifts the error to the low frequency area of the spectrum it is time to transfer it to a new grid, still coarser, and so on.

Two strategies form the core of the multigrid technique. The first of them solves a problem on a coarser grid to obtain an initial guess on the next finer grid. This is called ***nested iteration***. The idea is to find an initial guess with little computational effort, by means of building the initial guess on the next coarser grid. A basic principle lies under this idea: it is cheaper to find a solution if the initial guess is good, and nested iteration provides a good guess cheaply. The second strategy uses the residual equation to obtain an approximation to the error in the present grid. That approximation is computed by relaxing the residual equation on the next coarser grid. This version of the error is translated back to the current (finer) grid to be applied as a correction to the present representation of the solution vector. The advantage is that it is cheaper to work on a coarser grid. This second strategy is called ***coarse grid correction***.

The operation that involves transferring from coarser to finer grids is called ***interpolation***, while the opposite operation, i.e., transferring from finer to coarser grids, is denoted as ***restriction***. Although in the standard multigrid methodology the operators performing those processes are linear, in general interpolation and restriction could conceivably be nonlinear operators.

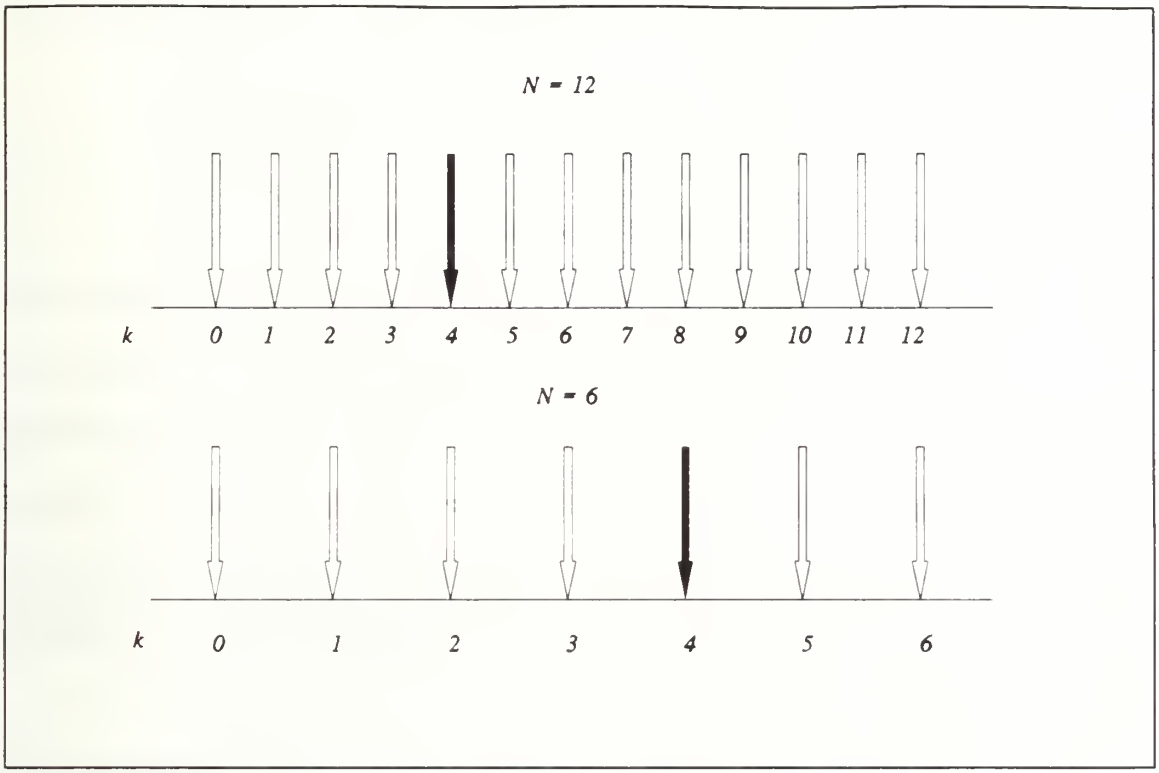

Figure 24.- *Spectral location of a k = 4 wave in grids with N = 12 and N = 6 respectively.*

The way one combines iterations at the different levels of coarsening yields several multigrid patterns, known as multigrid ***cycles*** or ***schemes***. The most common are the V-cycle, the μ-cycle and the Full Multigrid V-Cycle, described by Briggs (1987). We will briefly describe them next.

1. μ-Cycle

It is schematically represented in Figure 25. The operations performed are:

(a) Relax ν_1 times on the initial grid Ω^h with a given initial guess v^h; these relaxations (iterations) make the error $\mathbf{e}^h$ become smooth.

(b) Restrict the residual equation to the next coarser level Ω^{2h}. Since the structure of this problem is a parallel of the original problem, solve this residual equation

recursively, by a call to the μ-cycle. Do this μ times. As a result, we obtain an approximation for $\mathbf{e}^{2h}$.

(c) Interpolate to obtain an approximation of $\mathbf{e}^h$ on the fine grid Ω^h.

(d) Add the error $\mathbf{e}^h$ to $\mathbf{v}^h$ to obtain an improved approximation for $\mathbf{v}^h$.

(e) Relax ν_2 times the equation $\mathbf{A}^h \mathbf{u}^h = \mathbf{f}^h$ with initial guess $\mathbf{v}^h$ computed in (d).

Achieving the vector $\mathbf{v}^h$ by (a) through (e) is done much more rapidly than proceeding by (a) alone. This "speed-up" in the approximation process is obtained thanks to the third and fourth key facts described earlier in this chapter, that allow for working on a simpler grid (fewer points), thus making the computational work less. Concurrently, the oscillatory components of the restricted error are eliminated quickly by the relaxation steps. This is a crucial fact in multigrid.

The **V-Cycle** is a particular version of the μ-Cycle, for the special case that $\mu = 1$. When $\mu = 2$ the resulting scheme is called the **W-Cycle**.

2. Full Multigrid V-Cycle (FMV)

The FMV is depicted in Figure 26. The representation is taken from Wesseling (1992). Other authors (Briggs, 1987) consider the scheme starting at the coarsest grid (i.e. the initial coarsening to the coarsest grid is implied, and not shown). Briefly, the operations performed are:

(a) Restrict the original equation to all grids (in particular, to the coarsest).

(b) For $k = 1, 2, ..., M$ (coarsest grid)

Solve $\mathbf{A}^H \mathbf{u}^H = \mathbf{f}^H$ by performing a μ-cycle ν_0 times;

Interpolate to the next finer grid , i.e. from Ω^H to Ω^h (usually $h = H/2$);

Repeat until the finest grid is reached.

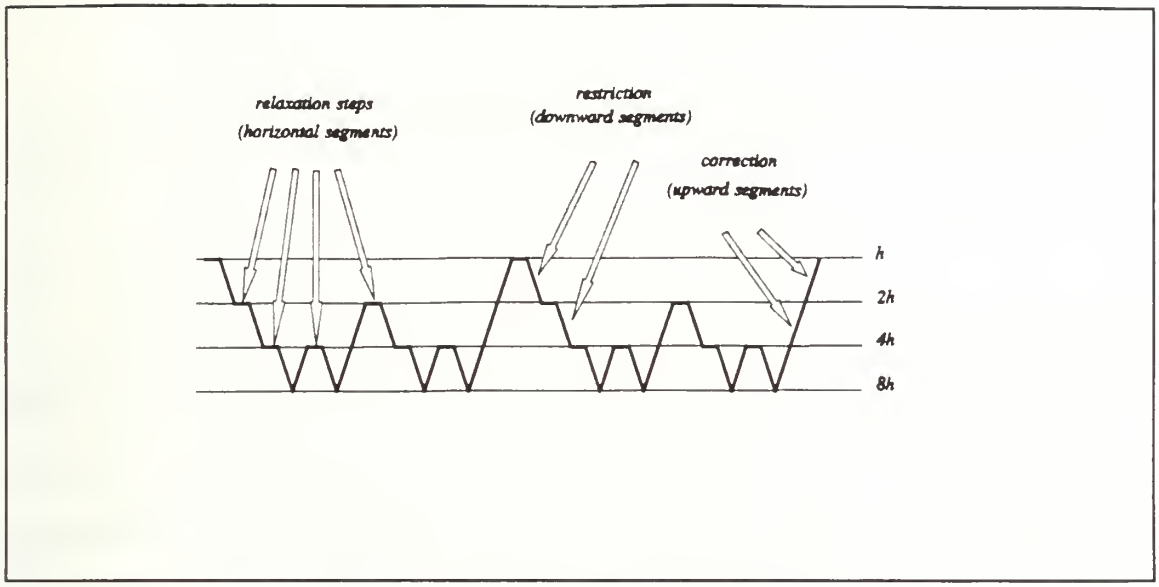

Figure 25.- *Schematic representation of the Multigrid μ-Cycle.*

This chapter has presented the principles and mechanisms of the multigrid method. We are now prepared to analyze the feasibility of such an approach to the solution of optimization problems.

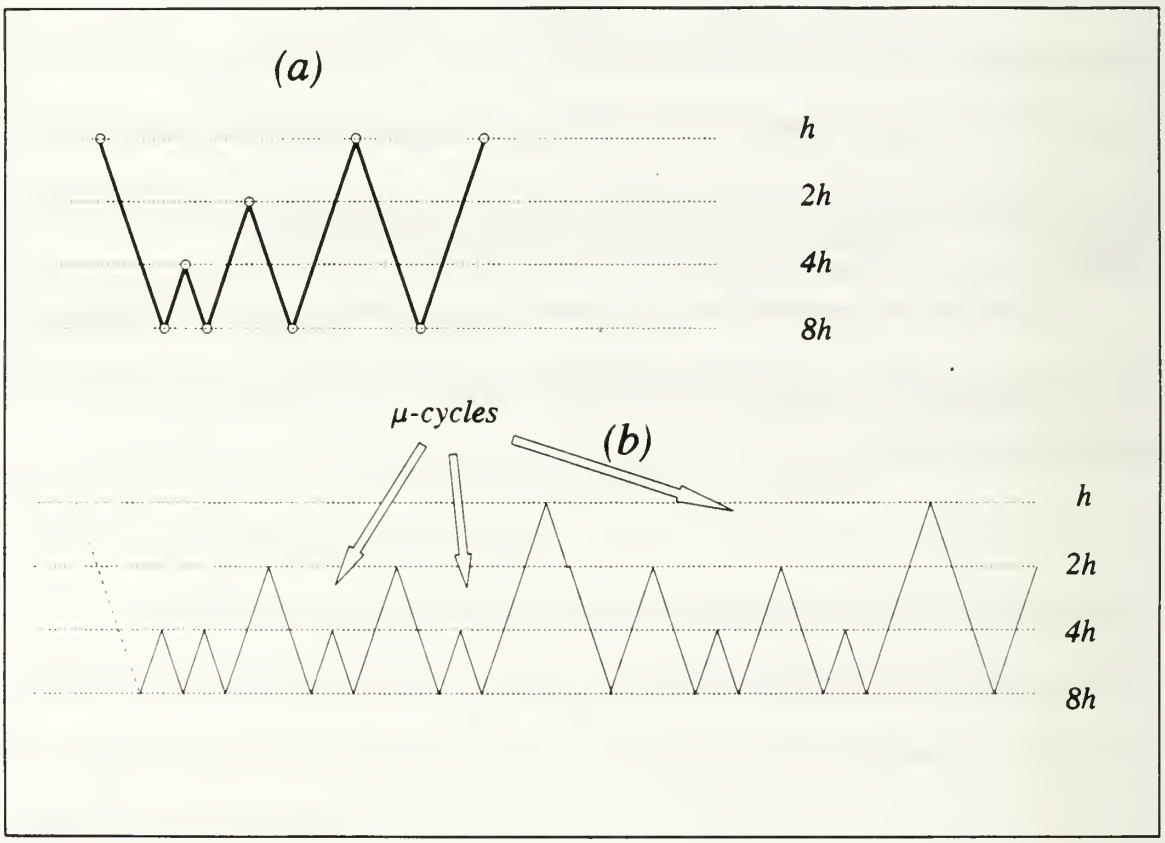

Figure 26.- *Full Multigrid Cycle. (a) $\nu_0 = 1$; (b) $\nu_0 = 2$ (not completed).*

IX. APPLYING THE MULTIGRID METHOD IN OPTIMIZATION: THE LONG TRANSPORTATION PROBLEM

A. INTRODUCTION

In Chapter VIII we have seen that multigrid is not a specific solution technique, but rather a philosophy of attacking problems that is general and powerful when applied to suitable problems. Specific solution techniques must be used. The restriction and interpolation operations are general concepts that should be defined in the context of each problem. It is the idea of changing the grid when the error is smooth that characterizes multigrid, as well as the idea that a solution in a "simpler" grid is a good starting point for attacking the problem on the next "finer" grid.

The next two sections list the characteristics of the types of problems that traditionally have been solved using multigrid, and a comparative list for the class of problems that typically are solved by optimization. Section D reviews some previous research. The objective in sections E and F is to extract the positive lessons from these analyses, point out the difficulties found and analyze them. Some insights into the multigrid method are included with conclusions that hopefully will aid the better understanding of the optimization procedures that could support the implementation of algorithms based on the multigrid methodology.

B. CHARACTERISTICS OF MULTIGRID PROBLEMS

Multigrid methods have been successfully applied to solve certain partial differential equations. These kind of problems, when solved numerically, are approximated by

systems of algebraic equations. Restricted to the linear case, we observe that these problems are characterized by the following common features:

- They usually correspond to *physical processes*, i.e., energy propagation, variation of space parameters when the subject is exposed to a system of forces, etc. In general, these type of problems describe phenomena in which there is a weak influence between pairs of local states corresponding to areas located far apart in the system.

- The system of equations that describe the phenomena usually express the influence of particular values of the solution (energy, speed, flow, etc) on a certain *neighborhood* of their space. In the example case of heat propagation through a rod, every row of the matrix **A** representing the system is obtained by a pattern of values that affect immediate neighbor components of the **u** vector (see chapter VIII). The zeros in each row of the matrix express mathematically the weak effect mentioned above, in the sense that interrelation between variables is restricted to the (neighbor) columns having nonzero coefficients.

  The weak global dependence between grid points determines its behavior through successive iterations. That is, given the state of the system, the local variation induced by a variable change is propagated as a sequence of local interactions, and the resulting state of the system is, in general, very close to the initial one.

- The problems are originally posed on a continuum. For computation purposes, the problem is discretized. The initial values of the variables, represented as vectors, constitute a sample of the continuous variables of the problem. Similarly, the solution vectors obtained are samples of continuous variables. For a sufficiently fine grid, if we express the values of the n variables at the grid points as n-tuples, the knowledge of the n-tuple corresponding to a grid point gives us some information about what the coordinates of the neighbor grid points could be. We can express this property, inherent in continuous functions, by saying that the problems have an ***implicit ordering*** in the composition of the solution vector. In Figure 27(a) a sample vector of x-values is represented. The dark area corresponds to the continuous set of x-values. The implicit ordering is materialized by the function f(x). This property is important for interpolation purposes.

- These problems are convergent when solved by iterative processes. That is, when the same problem is solved, using progressively approximated solutions, the error becomes smaller and smaller. This is typically expressed as the error being $O(h^p)$ for some p.

- As a result of being exposed to iterative procedures, the error term is smoothed. Thus, after some number of iterations little improvement is achieved, since smooth errors are very slowly reduced.

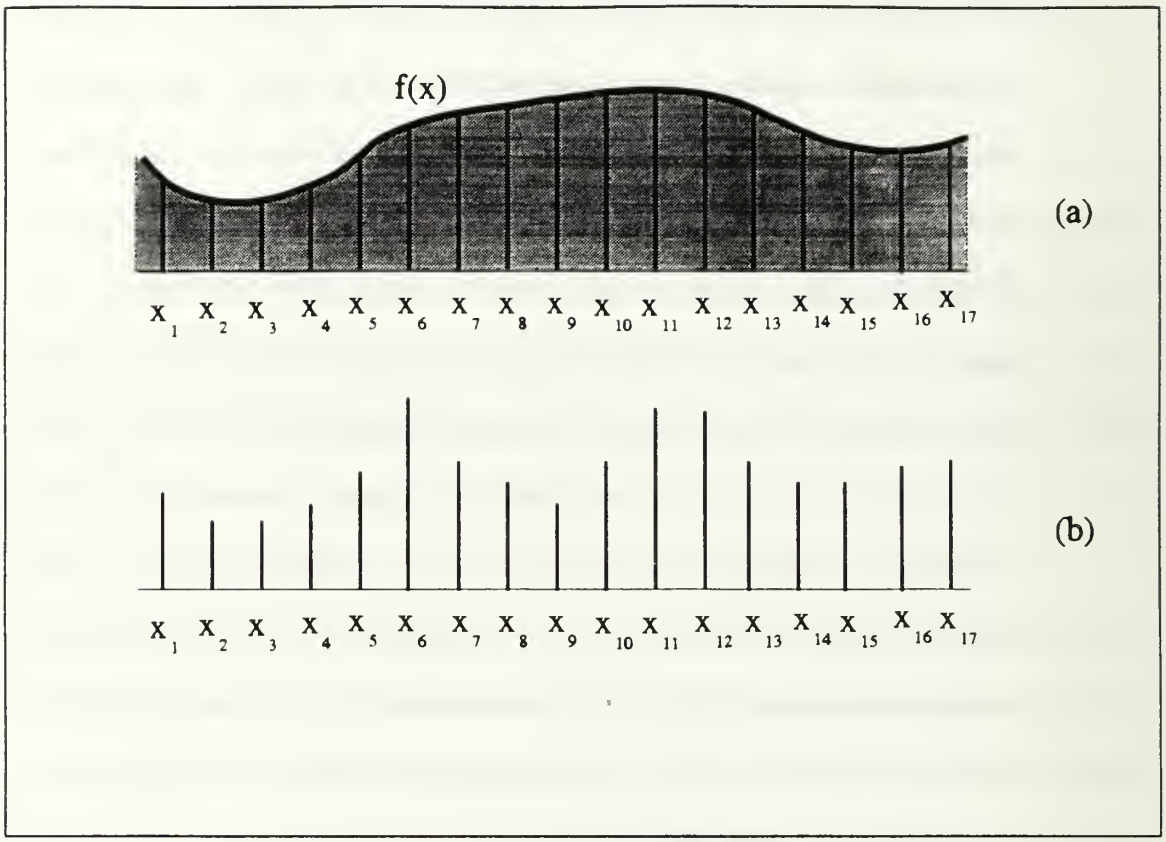

Figure 27.- ***X-Vector (a) PDE Problem; (b) Optimization Problem***

- Since the error is not available, the residual is used as a reference of the progress in the solution process. Although not necessarily the case, the residual is normally a good reference for this purpose.

- The restriction operator transfers the problem from a grid to the next coarser grid. This means that it needs to define the different aspects of the problem on the coarser grid. The interpolation operator performs the opposite operation. The fact that these problems possess a strong local coupling, together with the fact that the solution vectors are samples of continuous

sets, are key features for the interpolation process to work properly. In standard multigrid, both operators are linear.

C. CHARACTERISTICS OF (LINEAR) OPTIMIZATION PROBLEMS

The previous chapters have considered some optimization principles and techniques that could support a multigrid approach to solving (linear) optimization problems. Let us focus, for simplicity and consistency, on transportation problems.

- In general, they are constrained problems. They usually carry an associated set of linear constraints. This set is expressed as a matrix, and is used to check for the validity (feasibility) of the current solution. As the computation proceeds, the current solution vector solves a system of equations defined using only a submatrix (*basic submatrix*) of the constraint matrix.

- The objective of the problem is find a set of values for the variables that minimize (maximize) a function (objective function) of the n variables of the problem. There is no parallel concept to that of the residual in the solution of the systems of equations. On the other hand, since the main goal is maximize (minimize) the objective function, there is a valid reference (at hand) of how close we are to the solution as the process proceeds. This is because, at each step, bounds for the optimal value of the objective function can be calculated, providing a means to measure that proximity. (Notice that usual optimization algorithms, i.e., simplex method, keep track of the best solution obtained and bounds. They proceed "improving" the current objective, so that

as more steps are performed, the bounds on the objective function are narrower).

- Contrary to the PDE case, the current solution (including in this term both initial and final solutions) is represented by a vector which does not constitute a sample of a continuous set of values. This is schematically presented in Figure 27. The set of values of the variables in an optimization problem is (a) finite and (b) is formed by all the possible data points.

- In general, there is no relation of proximity determining the structure of the equations. That is, contiguous sources or sinks are not necessarily correlated, they do not necessarily influence each other. Compared to the PDE case, we can say that, as a general rule, if we knew the value of a flow component x_i, this says little about the values of "contiguous" components x_{i-1} or x_{i+1}, for example. There is no implicit order materialized in the form of a relationship like f(x) in the continuous case (a).

- When the problems are bounded, they have an optimal solution. The optimal solution may or may not be unique. Optimization techniques yield solutions generally based on an enumeration process. It is the finite structure of the problem which guaranties the solution to the problem.

- In most practical problems, each variable has individual bounds. Normally, these bounds are locally assigned, and constitute an essential parameter in determining the problem behavior when being solved.

D. PREVIOUS WORK

Ron (1987), proposed the idea of a multigrid approach for the N-city Traveling Salesman Problem (TSP). This is a standard, NP-complete problem that arises in several contexts. For a description see Roberts (1984). An estimate of the optimal length of the closed TSP path L for this case is known to be equal to

$$L = 0.749\sqrt{N} + o(\sqrt{N}) \tag{9.1}$$

for sufficiently large values of N (Bonomi and Lutton, 1984). The Asymptotic TSP (ATSP) finds paths within a few percent of the above value. In her research, Ron considers a number N of cities randomly (e.g. in a uniform distribution) positioned in the unit square. Instead of taking an arbitrary first approximation (random order of cities), she constructs an approximation as follows. Coarsen the problem by creating a sequence of $\lceil N/2 \rceil$ pairs of cities (referred as *coarse* cities), each being located at the mean positions of the two cities it represents. The pairs are constructed by associating the nearest available one with each city in turn. Her method was tested in a variety of examples and gave reasonable first approximations, yet it failed to become a better solver since the coarser problems are not, in general, geometrically equivalent to the finer problems, and thus the solution interpolated from a coarser level is not necessarily a solution for the finer one. Still, for the ATSP she reports results with paths no longer than 30% off the optimal solution.

There are some characteristics in the ATSP problem that makes it attractive to being approached via multigrid. As the number of cities grows, the minimization processes at regions of the plane far apart are weakly coupled to each other. Surely, the minimized path will not jump back and forth between cities far distant from each other, i.e., it is expected first to visit neighbor cities before switching to other far regions. This implies the assumption of neighborhood influence, described in section A. That is, the cost of visiting remote cities before nearby cities must be relatively large. In such conditions, the reported results are logically explained. However, from the optimization theory standpoint, the computational results cannot be considered attractive. Ron reports better results in applying multigrid to a physics problem, the Ising spin problem, which is the core of her thesis.

Kaminsky (1989) proposes an algorithm for the long transportation problem using multigrid techniques. Presumably, the selection of the long transportation problem for experimentation comes from the fact that origin nodes, being small in number, are left intact, i.e., they do not change through the process of being exposed to the different grids. This simplifies the problem. He defines the problem as a ***geometrical transportation problem***, because of the additional assumption that origins and destinations have locations in space, and the cost of shipping between any pair of origin-destination nodes is related to the distance between them. (The idea of intrinsic ordering appears in the setting of this problem, allowing it to be posed as multigrid). Kaminsky uses aggregation to define the coarse destinations. Two neighbor nodes on grid h are aggregated into a single node in grid 2h, with demand equal to the sum of the two demands. Supply nodes are not aggregated. Two different rules are used to set the costs in the coarser problem.

The first rule is a "***weighted aggregation***" (Zipkin, 1977), which provides the best initial solution, compared with other schemes. The second rule, valid for subsequent steps, makes use of a previous state of the problem. In this rule, the costs to coarser destinations are the dual solution values associated with each origin node when solving the associated ***block problem***. This block problem is defined for a single coarse node. There are as many block problems as coarse destination nodes at each level. The interpolation process (coarse-to-fine grid) is the solution of the block problems themselves. In Figure 28 there is an interpretation of an individual block problem. The values X_{iJ} are the flows computed when solving the coarse problems, i.e. flow from supply node i to coarse destination J.

Of special interest is the discussion that Kaminsky provides on local relaxation in Chapter 5 of his thesis. A process similar to that of coarse to fine interpolation is applied to a subgroup of destinations to "improve" the current feasible solution at a determined state of the problem. This involves solving a linear programming subproblem. The reports of the experimentation are that, by applying such method of relaxation, the results are ineffective in improving a solution in which the origins supply sets of destinations that are contiguous in the space. But this is exactly the expected character of the optimal solution (*regionalized* solution). The converse is true when the solution is not regionalized. The best way to do local relaxation is left as an open question. Even so, his reported results can be considered an important step towards multigrid-based optimization solving techniques.

Cavanaugh (1992) removes the spacial dependence of shipment costs in an attempt to generalize the problem proposed by Kaminsky. This extension is important, since many

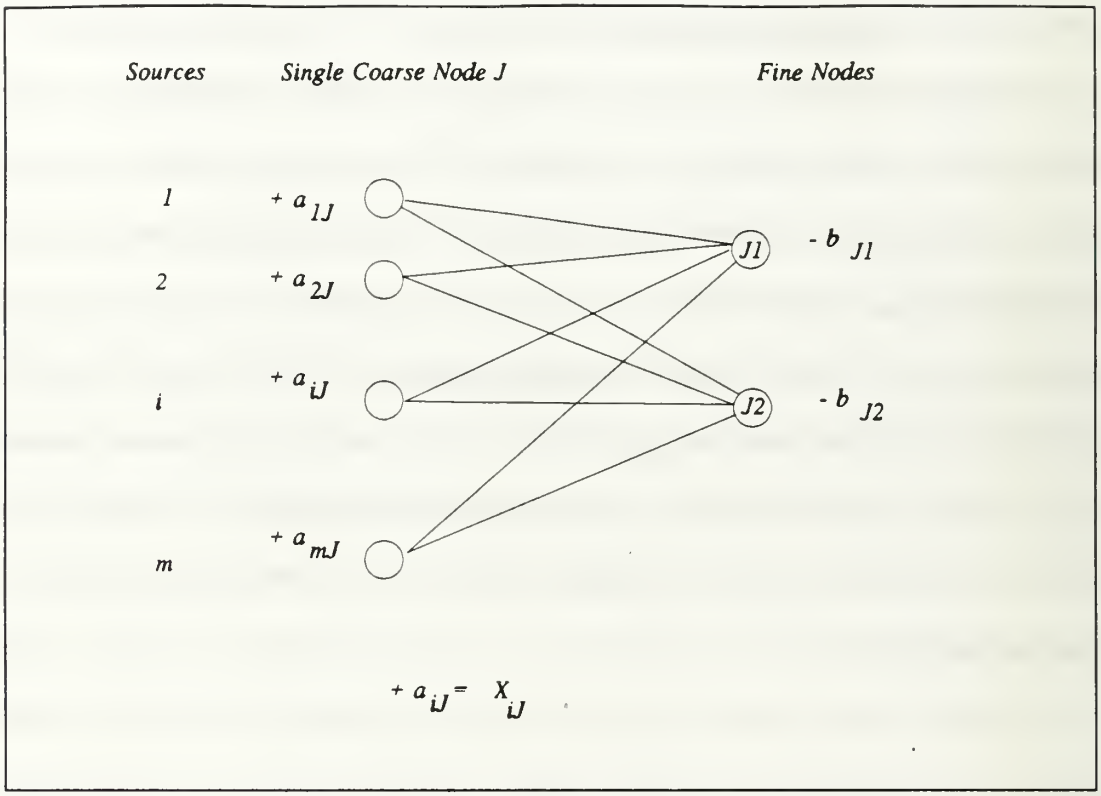

Figure 28.- ***Kaminsky's Block Problem for Coarse Destination Node 'J'.***

of the problems - a large majority in real-life - that can be formulated as transportation problems have no physical space interpretation. The addition of graph-theoretic properties to the multigrid components makes his work attractive. He considers the proximity of nodes in the cost space, which we define as follows:

Given a transportation problem (TP), consider the set C of m-tuples $(c_1, c_2, ..., c_m)$, where the c_i's are all the possible costs that could be assigned to the shipment of a unit of commodity from origins 1, 2, ..., m, respectively. The set C is referred to as the ***cost space*** of the transportation problem TP. We can also say that TP is defined on the cost space C. The cost space is an m-dimensional euclidean vector space. For practical purposes, sometimes we restrict the c_i's to be integers and, sometimes, to nonnegative numbers. For each particular transportation problem the set of arc costs is a finite subset

of its cost space. Each destination node j has an associated point in the cost space defined as $(c_{1j}, c_{2j}, ..., c_{mj})$. The k^{th} coordinate of destination node j is the the cost c_{kj} of shipment to it from supply node k.

Cavanaugh, like Kaminsky, uses *aggregation* of nodes (in the cost space) to design the problems associated to the coarser grids. Also as in the previous work, there is no limit on the capacity of the arcs defining the associated network. The way nodes are selected to be aggregated is as follows: the demand nodes are first sorted by increasing cost of shipping from supply node 1, and divided into two groups about the median of the sorted cost (Figure 29(a)). Each of the resulting groups are then subdivided into two groups, the criterion now being the cost of shipping from supply node 2 (Figure 29(b)), and so on. For simplicity, Figure 29 represents a problem with only two supply nodes. Each group created in this way is then sorted by the same method, generating smaller groups. The process is repeated until groups of size 2 are all that remains.

The interpolation process solves similar problems as those mentioned in Kaminsky's thesis. But Cavanaugh makes use of the tree structure of basic solutions and the special characteristics of the transportation problem to speed up the calculations. Also he introduces the use of reduced costs as a measure of optimality.

The local relaxation process is by cycle-removal. When interpolation is performed, a set of local problems is solved. These problems produce locally optimal solutions. Since all the local problems share the same set of supply nodes, when constructing a global solution from the local solutions a certain overlapping takes place. This causes the graph of the global solution to have cycles. Cycles must be eliminated for the solution to be optimal. To check for cycles, two adjacent Mx2 local problems are chosen and the flows combined to form a solution to an Mx4 "regional" problem. Cycles are sought and

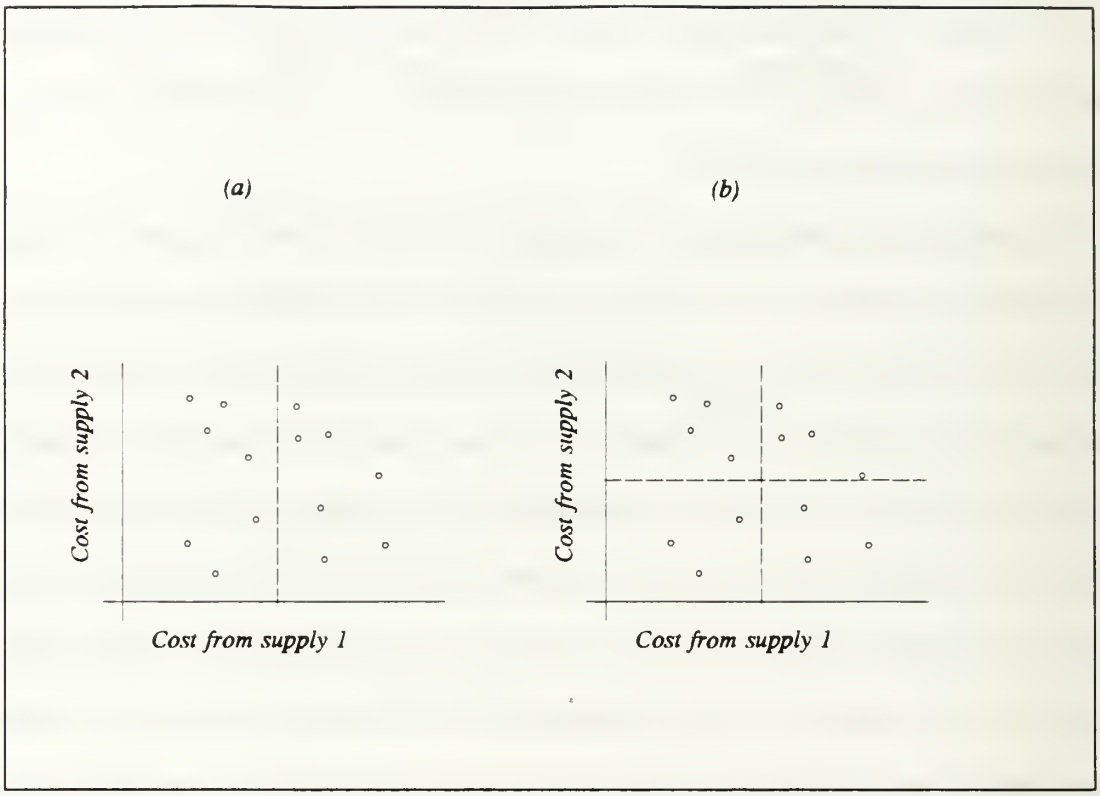

Figure 29.- ***Cavanaugh's coarsening of Demand Nodes***

eliminated. This produces a local relaxation scheme. The mechanism for the cycle removal is inspired by the one used in GNET (Bradley *et al*, 1977). Cavanaugh's multilevel algorithm performed well on problems with only two or three supply nodes. His solutions on problems consisting of 3 supply nodes and 1024 demand nodes are 8.41% above optimality using cycle removal. For 5 supply nodes and equivalent problem size, the results are at least 58% above optimality, which cannot be considered very promising.

Cornett (1993) continues the research on the long transportation problem. Node aggregation in cost space is the selected mechanism for restriction. In Cornett's work, regular grids in cost space are considered for a problem consisting of three supply nodes. With this scheme, nodes are grouped together with other nodes lying close each other in

absolute terms. This way, the influence of local variations is more "controlled", in an attempt to restrict the effect of variations to the close neighborhood. Multigrid methods tend to perform well in such conditions. In the regular grids presented by Cornett, once the mesh size is determined for a grid, the cost space is partitioned in elementary polytopes, determined by intersections of the coordinate planes. The points of the grid are the vertices of the elementary cells. The cells surrounding each grid point define an elementary cell in the next coarser grid, and so on. Notice that the cost space is also partitioned in this coarser grid (no overlapping occurs), so not all the grid points are centers of cells in the next grid. Figure 30 represents a regular grid, formed by the small circles. In that figure, two filled and larger circles represent points in the next coarser grid. Notice in the figure that grid points located on each side of a cell are shared by two consecutive cells. Also those located on each edge are shared by four, and those on each vertex are shared by eight cells.

The restriction operation defines the demand of a point in the coarser grid by the weighted average of the corresponding *cell*-demands (denoting by cell to the set of points included in the elementary cube centered at each coarse point).

The interpolation process divides the flow on a coarser node "proportionally" among its corresponding cell demands in the finer grid. The flow coming from each of the supply nodes is considered as the set of supplies, so defining a coarser node subproblem. The global solution is then obtained combining the solutions obtained from these *local subproblems.*

The relaxation procedure used in this approach consists of a check that every demand node has been served in the cheapest possible way. For those flows not passing this test, a pair of temporary supply-demand nodes are created. The supply-demand

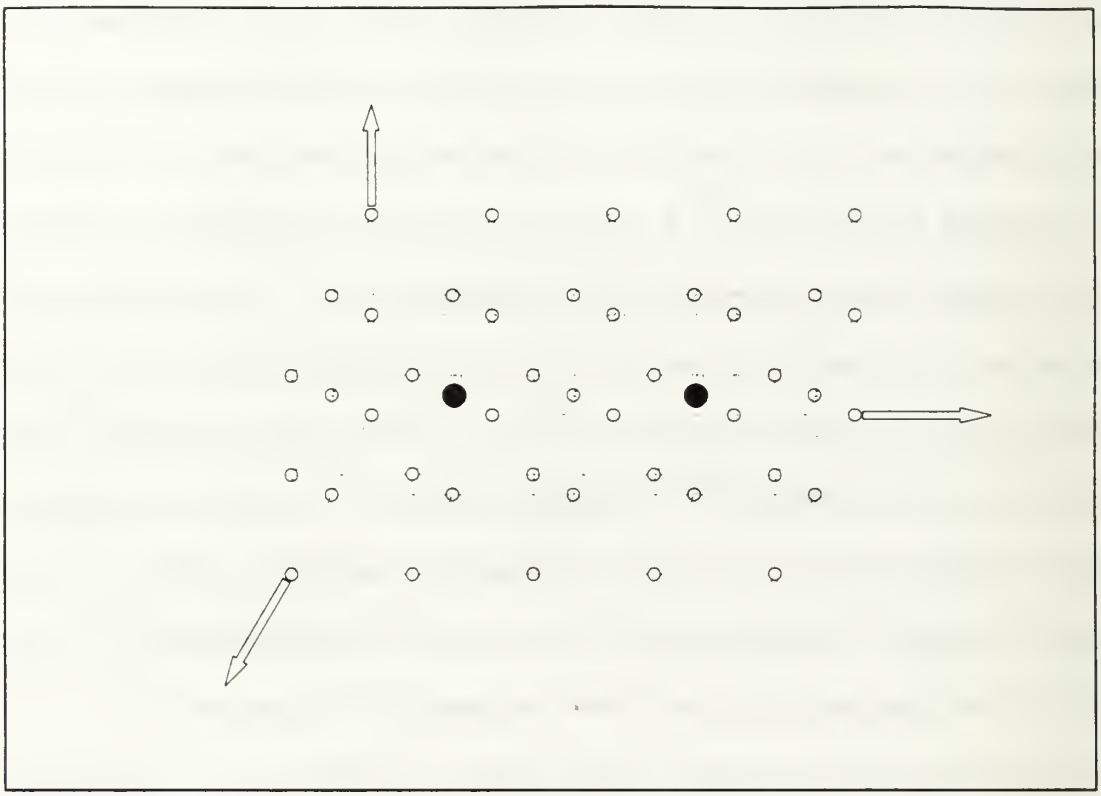

Figure 30.- ***Three Supply Nodes Problem (Cornett).***

values of these pair of nodes corresponds exactly to the flow that has been determined to be non-adequately distributed. Each created pair is then associated with a flow in the current global solution. This flow is removed from the current solution. When all such pairs of temporary nodes have been created, the complete set of temporary nodes, and their corresponding supply and demand values, define the flow that, in the current solution, is considered to be not-optimally allocated. A new transportation problem involving only those flows is created and solved, and its solution combined with the part of the flow optimally distributed, so obtaining a new current solution in the relaxation process. Actually, a maximum of only one node is created for each of the existing nodes, and the supply (demand) of that node increased by the value of each non-optimal flow. Therefore, at each step, the original problem is divided into two parallel problems, one containing the

part of the flow vector considered "optimal", and the other containing that part of the flow vector considered "not optimal". As a result, the relaxation process improves the flow at each iteration.

By using the above scheme, Cornett designed a V-cycle type algorithm that solved problems with 3 supply nodes and up to 29,791 demand nodes within 4% of optimality, using up to 5 grid levels in the design scheme.

E. RESEARCH OBSERVATIONS

At this point, it is useful to reflect on the research process in order to extract some positive lessons and to point out other possible causes of problems. This is in fact one of the objectives of the present work.

Two facts deserve special attention when studying the works described above. They will be considered independently.

1. Node Aggregation

Node aggregation has been viewed as the natural way of restriction. It presents the following advantages: (a) the coarsened problem is the same type of problem, and (b) has fewer nodes. The first property is typical of multigrid contexts. And the second, due to the combinatorial nature of optimization problems, reduces effectively the computational effort needed to solve the coarsened problem.

For node aggregation to be suitable for multigrid use, it should maintain the general property of multigrid: local effects must propagate weakly. In this sense, aggregation of nodes causes two conflicting effects. On one side, grouping of neighbor nodes maintains a hierarchical structure when going from grid to grid. On the other side,

the way nodes are grouped may reinforce global influence. This is illustrated in the following example.

Consider a simple transportation problem with two supplies and eight demands (Figure 31). When grouping nodes following the rule used by Cavanaugh one would arrive at the groups depicted in (a), in opposition to the more natural groupings of (b).

Kaminsky uses two different rules for node aggregation. In one algorithm, the total area of the problem is divided into two "sub-blocks" by a line in the direction of the x-axis. Each one of these sub-blocks is then divided by a line in the direction of the y-axis, and so on. This could also lead to situations parallel to that depicted in Figure 31, but in the physical space.

A different approach is proposed by Kaminsky in his second algorithm. Here a "bottom-up" blocking is achieved by constructing a graph whose vertices are the destinations in the fine grid. Each destination is connected to the nearest four destinations in space. Then the destinations are blocked by finding a maximum cardinality matching. There is nothing that prevents extreme matchings like that in Figure 32(a), hiding the natural structure of the problem, shown in Figure 32(b).

2. Regular Grids

Regular grids appear to better deal with the circumstances described in the previous paragraphs. But they also present some disadvantages.

Setting the long transportation problem on a regular grid tends to preclude the pairing of distant nodes (in cost space). Furthermore, the local effects affecting the state of the problem tend to propagate in a more "controlled" way, since the area of immediate influence of a node consists of a group of nodes which do not necessarily exist in the

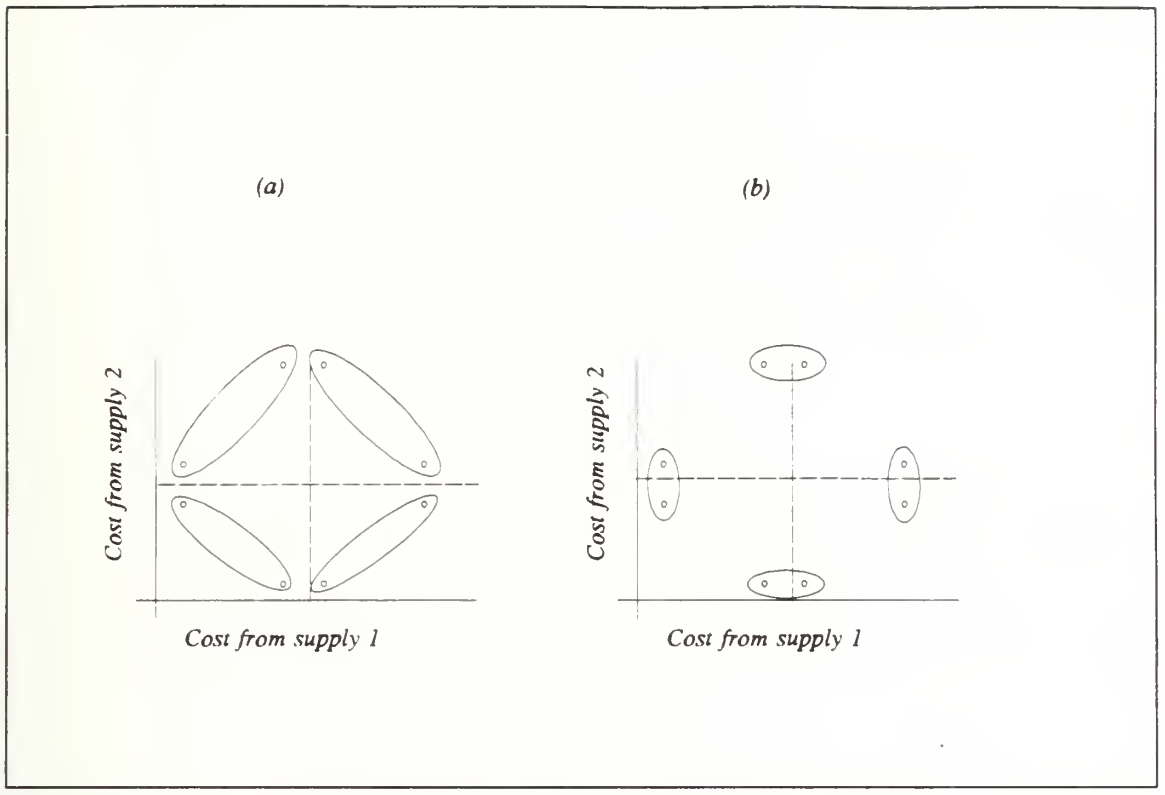

Figure 31.- ***Two different Aggregations of Nodes in Cost Space.***

actual problem. It could be said that regularity is used in an attempt to force predictable behavior.

We can describe the effect of a regular grid in an intuitive manner observing Figure 33. The original problem (a) is shifted to a regular grid (b) that is finer than the original setting (which now becomes a coarse grid). As a consequence, the gaps of the flow vector **x** in (a) are regularly filled with intermediate flows in (b), making the vector **x**'s costs "appear" as a sampling of cost locations of a more populated cost space.

In normal sampling (the PDE example), the magnitudes of components of the **x** vector tend to stay around the corresponding values in the continuous domain. On the other hand, in the case just described above concerning regular grids, there is a need to combine neighbor values to obtain the value of the "sample" in the next coarser grid. One

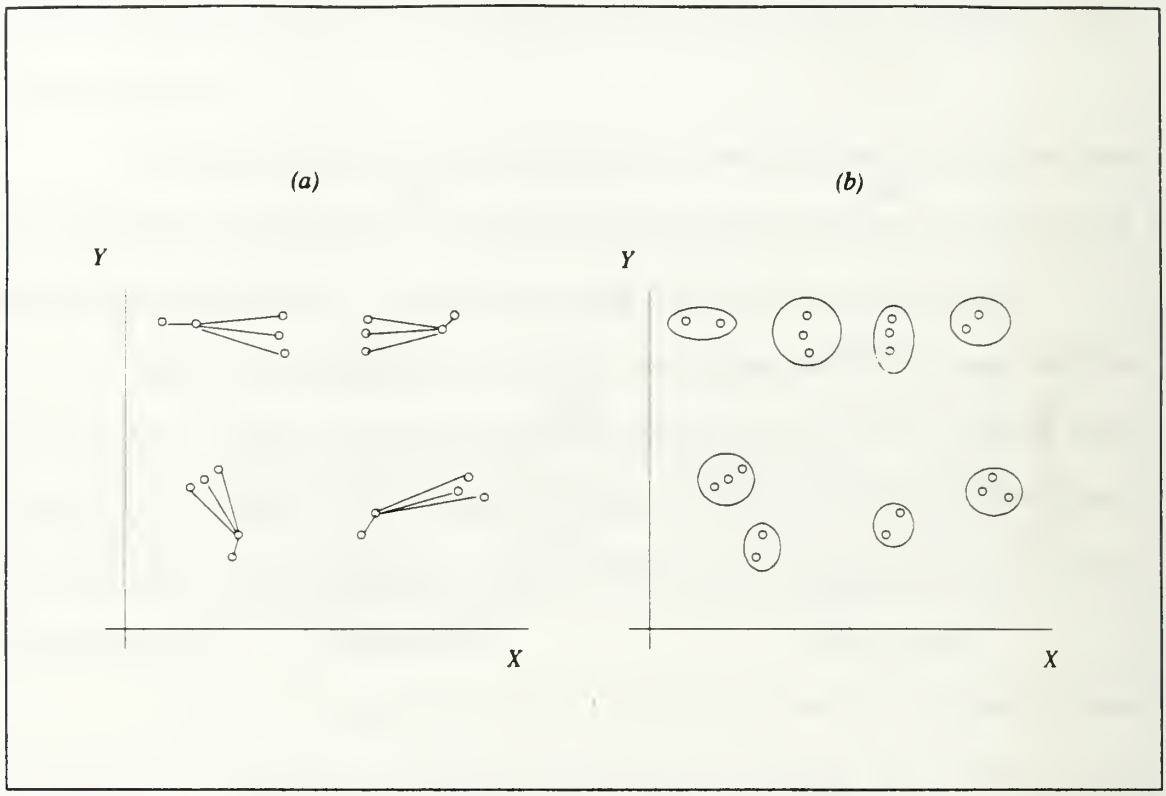

Figure 32.- ***"Bucket" Aggregation of Nodes (Kaminsky).***

can imagine the difficulty of devising a sampling process (in optimization) such that the values of flows in successive grid representations tend to converge, when performing node aggregation in the terms of the previous approaches.

Since the dimensionality of the problem equals the number of supply nodes, we can imagine that the rapid growth in size of the problem, already pointed out (Cavanaugh *et al*, 1992) for irregular grids, will be much more important in the regular grid case. This limits the practical field of application of regular grids to problems with only a few supply nodes.

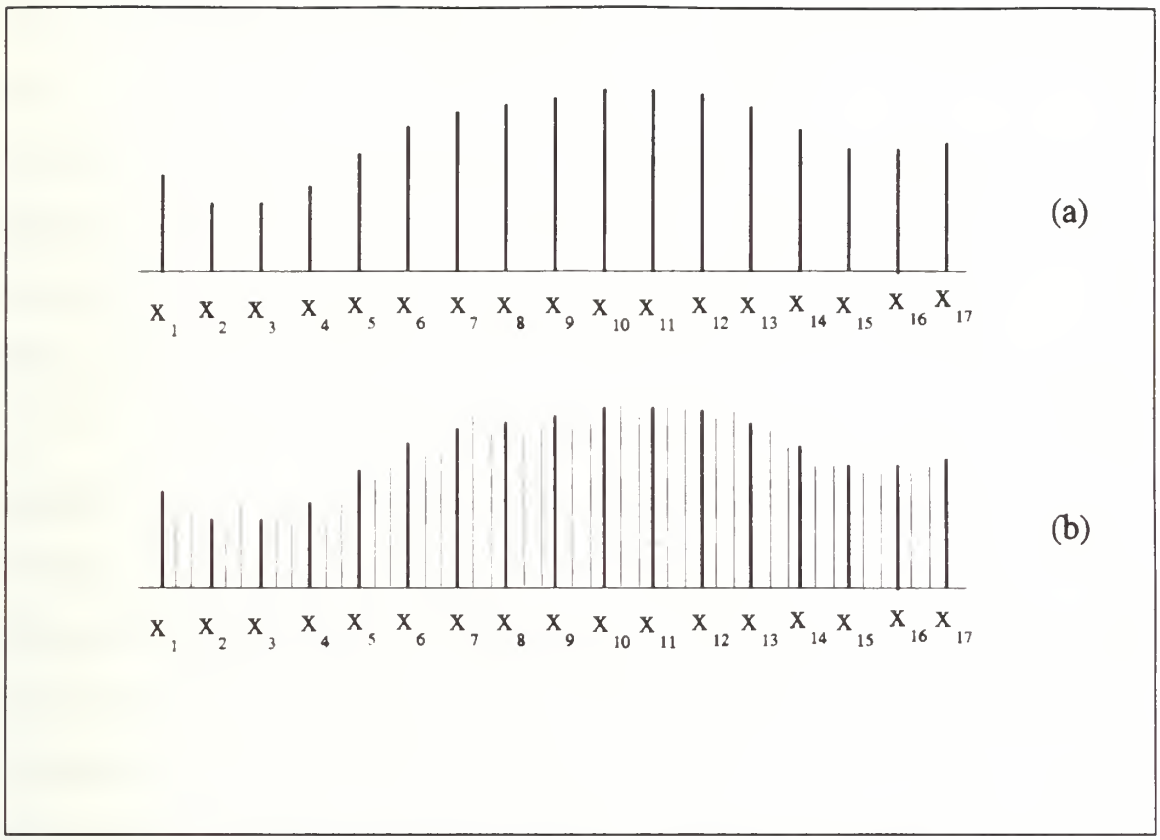

Figure 33.- *Analogy of Regular Grid and Continuous Sampling.*

F. CONCLUSIONS

This final discussion connects some of the optimization tools presented in the previous chapters with the work that has been developed towards the implementation of multigrid methods in optimization. The conclusions are an attempt to compile the limited experience in applying multigrid ideas in optimization with the specific optimization techniques that could play a relevant role when designing multigrid components.

Three aspects will be considered. First is the general question of whether multigrid should be applied to optimization, or vice-versa. Second, applications of multigrid approaches to optimization problems will be suggested. Third, the various optimization

techniques presented in previous chapters are connected to the multigrid process, with some insights and recommendations for future research.

1. Multigrid towards Optimization versus Optimization towards Multigrid

Throughout the present work, this question is a constant: (a) should multigrid be used as a general way to approach problems, transferring the idea to the solution of optimization problems, or (b) should optimization techniques be directly transferred to the multigrid scheme and so build a multigrid scheme with optimization blocks.

The conclusion of this thesis is that the multigrid method is more a philosophy of attacking problems than a specific solution technique. Multigrid concepts can be applied in optimization to develop a strategy for approaching problems. But not all the components of a multigrid scheme are directly transferable to optimization, where problems, in general, are structured so that local variations typically propagate widely into the global problem. Nevertheless, techniques and procedures of optimization can be used to handle some of the troublesome properties that these problems present to a multigrid approach. This is covered in the following points.

2. Field of application

The previous point leads to the consideration of how general an approach multigrid is for optimization problems. The study of previous work in section D, led to the observations in section E, from which it follows that aggregation of nodes is a natural process to be used as a restriction operation. The experience, however, is not as promising as it appears at first glance. The associated difficulties explained in section E.1 suggest that only problems with a suitable structure are amenable to node aggregation as a restriction operation. Problems structured so that clusters of nodes can be identified

are specially oriented to a multigrid approach. So, ***clustering-type*** of layouts are expected to produce better results. For example, in cost space, the problem represented in Figure 34(a) has two identifiable clusters of nodes, with each one formed by two pairs, that can be thought of as clusters at a different level. That kind of layout is expected to be more successfully solved in a multigrid fashion that the spread layout in Figure 34(b). Often, when experimenting with algorithms, one tends to randomly construct problems. This methodology is more likely to produce designs like Figure 34(b). The conclusion here is that experiments must be designed carefully when node aggregation is the method chosen for restriction. The number of grids that can be implemented is determined by the separation between clusters. In general, the distance between two members of a cluster must be small compared to the distance between two clusters. Furthermore, a new level with its corresponding grid should be introduced only if the clusters can be partitioned into groups, such that the distance between groups of clusters is large when compared to the distance between individual clusters within a group. This process, when iterated, determines the maximum number of grids that a problem can have. Therefore, the maximum number of grids that can be introduced is determined by each particular instance of a problem.

A characteristic to examine in order to determine if a problem is suitable for multigrid approach is whether there exists a weak relation between points of the grid located far apart. This can be expressed, for instance, in terms of a function assigning costs or other problem parameters according to a rule. An example is the case of Kaminsky's geometric transportation problem. In general, it is desirable that a mapping could be found so that some of the main defining characteristics of "local" points of the problem could be gathered together and be given in the form of a function. Kaminsky's

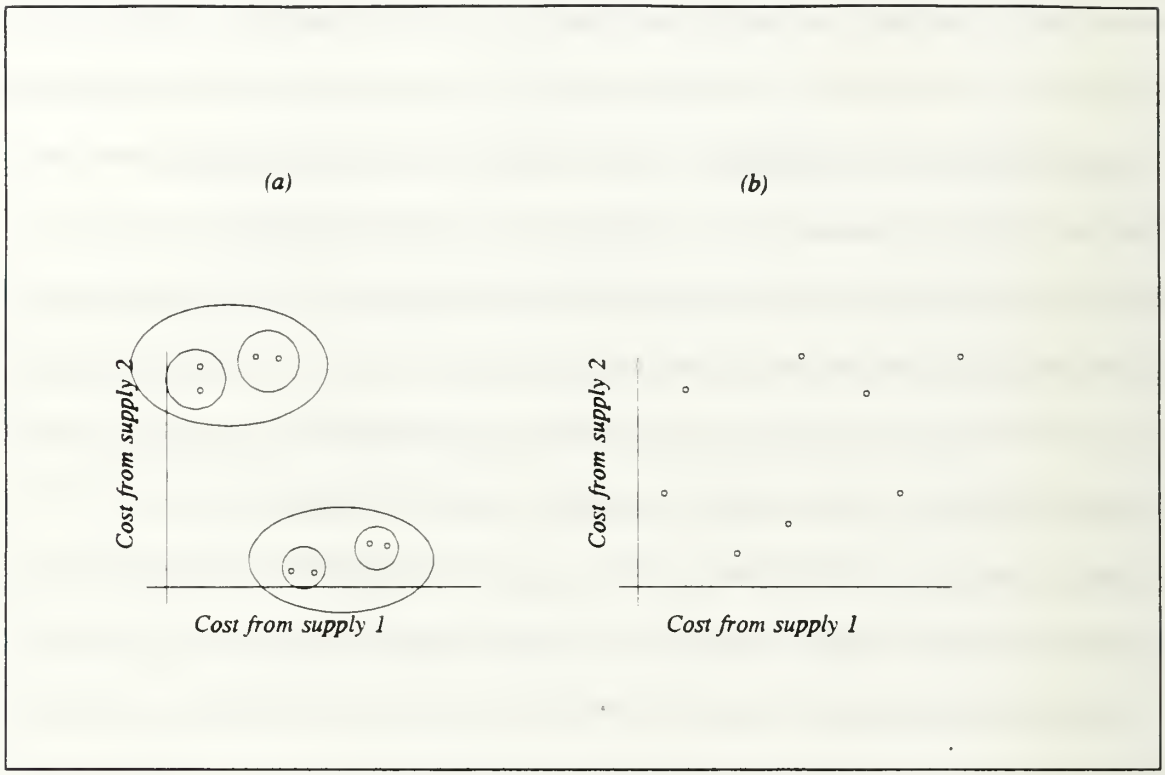

Figure 34.- ***Clustered versus Spread Structure of Problems.***

geometric transportation problem, for example, defines a map between locations in the space and costs. With that type of problem design it is possible to check for a cluster structure favoring multigrid, and determine the depth in levels that can be expected.

Demands play an important role when aggregating. It is conceivable to increment the dimension of the cost space or the physical space to account for the demands of (destination) nodes. Another property that could be considered for aggregation is the likelihood that a demand node be supplied by a given source. This can be defined in terms of available supply, required demand and reduced cost. In general, it is unlikely that a perfect mapping could be found defining a preference ordering. Finding such mapping is probably as difficult as solving the problem itself. Occasionally, a few properties or parameters have a stronger influence than the rest in determining supply

policies, and it is in such cases when the structure of the problem can be expected to support a multigrid approach.

The uncapacitated long transportation problem possesses some individual characteristics that make it suitable for a multigrid approach. In Chapter III.C we mentioned the fact that an optimal extreme point solution to a transportation problem has at most m-1 demand nodes that will be supplied by at least two supply-nodes, while the remaining at least n-m+1 demand nodes are supplied by only one supply-node. The implication is that, since n is much larger than m, the number of destinations served by only one origin is relatively high.

3. Unconventional Grids - Decomposition

The difficulty with regular grids pointed out in section E.2 raises the question of a less conventional interpretation of the classical concept of grid for optimization problems.

In his article "Levels and Scales", Brandt(1985) gives the following illustration of the favorable characteristics of problems where multigrid succeeds. A two-level hierarchical structure should operate in the following way. The global government first gathers some general data summarized at the local level, representing the sum totals of local needs, important overall constraints, etc. Based on these it prepares preliminary global plans. These global plans give the local governments the framework for devising their own, more detailed plans. These global plans do not quite fit the local situation and, therefore, need some adjustments or corrections. So, in a second round, the global government again gathers information summarized at the local level, now representing sum totals of needed *corrections*. Since in practice this process is seldom fully

recognized, let alone fully effectively organized, more such rounds may be needed. When more levels of government are involved, the process is applied recursively, in a variety of manners.

The ideal operation of the two-level hierarchical government is fully analogous to a two-grid process. The problem is first represented on the coarser grid (e.g., by averaging its equations to the scale of that grid). The (approximate) solution to the resulting coarse-grid problem, once computed, is then interpolated to the fine grid, serving there as first approximation, a framework, to be next improved by fine-grid processes, such as relaxation. This fine grid processing finds the fine features of the solution which were invisible to the coarser grid, and also, as a result, encounters some residuals of global (smooth) errors, which it cannot effectively reduce. So, in the next round, the residual problem is approximately transferred, by some averaging, to the coarse grid, where it can be efficiently solved and its solution is then interpolated back to the fine grid and added as a correction to the previous fine-grid solution. This is a two-level full multigrid (FMG) algorithm.

At this point, let us analyze and compare the two-level work of a decomposition procedure, like those described in Chapter IV. Consider a large system that is composed of smaller subsystems 1, 2, ..., T. Each subsystem i has its own objective, and the objective function of the overall system is the sum of the objective functions of the subsystems. Each subsystem has its constraints designated by the set X_i , which is assumed to be bounded. In addition, all the subsystems share a few common resources, and hence the consumption of these resources by all the subsystems must not exceed the availability given by the resource vector **b**.

Recall that the economic interpretation of the dual variables (Lagrangian multipliers, **w**) represents the rate of change of the objective as a function of b_i . Hence - w_i can be thought of as the price of consuming one unit of the i^{th} common resource. With this in mind, the decomposition algorithm can be interpreted as follows (Bazaraa *et al*, 1990). With the current proposals of the subsystems, the superordinate (total system) obtains the optimal weights of these proposals and announces a set of prices for using the common resources. These prices are passed down to the subsystems, which modify their proposals according to these new prices. A typical subsystem i solves the following subproblem:

Maximize $(\mathbf{w}\mathbf{A}_i - \mathbf{c}_i)\,\mathbf{x}_i + \alpha_i$

Subject to $\mathbf{x}_i$ in the set X

or equivalently

Minimize $(\mathbf{c}_i - \mathbf{w}\mathbf{A}_i)\,\mathbf{x}_i - \alpha_i$

Subject to $\mathbf{x}_i$ in the set X

The original objective of the system i is $\mathbf{c}_i\,\mathbf{x}_i$. Note that $\mathbf{A}_i\,\mathbf{x}_i$ is the amount of common resources consumed by the i^{th} proposal. Since the price of using these resources is **-w**, then the indirect cost of using them is $-\mathbf{w}\mathbf{A}_i\,\mathbf{x}_i$, and the total cost is $(\mathbf{c}_i - \mathbf{w}\mathbf{A}_i)\mathbf{x}_i$. Note that the term $-\mathbf{w}\mathbf{A}_i\,\mathbf{x}_i$ makes proposals that use much of the common resources unattractive from a cost point of view. The mechanism is the following.

Subsystem i announces an optimal proposal $\mathbf{x}_{ik}$. If this proposal is to be considered, then the weight of the older proposals $\mathbf{x}_{ij}$ must decrease to "make room" for this proposal; that is $\Sigma_j \lambda_{ij}$ must decrease from its present level of 1. The resulting saving is precisely α_i . If the cost of introducing the proposal is less than the saving realized, then the superordinate would consider this new proposal. After all the subsystems introduce their new proposals, the superordinate calculates the optimum mix of these proposals and passes down the new prices. The process is repeated until none of the subsystems has a new attractive proposal; that is, when $(c_i - wA_i)x_{ik} - \alpha_i \geq 0$ for each i.

The case of decomposition resembles, although is not exact, the case of the two-level hierarchical structure described above. The process of decomposition, conveniently helped by a mechanism to discover "easy" sets of constraints, gives a method to detect hierarchical structures in problems where they are not initially explicit. Here the conventional concept of grids is replaced by the more general concept of level, in order to facilitate a mapping of the problem into a somewhat hierarchical structure.

4. Unconventional Grids - Scaling

We can consider scaling as a multilevel technique. Here we will establish the parallels between the scaling techniques and the multigrid approach. Recall, for example, cost scaling. Here, the basic mechanism is work on a state of $\varepsilon/2$-optimality for each value of the scaling parameter ε.

Recall (Brandt, 1985) that the discretization error, i.e., the difference between the true solution of the differential problem (in the PDE case) and the exact solution of the discretized equations, has a relative magnitude clearly determined by the relative magnitudes of the discretization mesh size and the solution scale. Thus, exactly those

errors that are slow to converge by relaxation processes on some grid can be approximated on a coarser grid, where the mesh size is comparable to their scale and hence their convergence need not be slow.

In the scaling procedures, i.e., cost scaling, the scaling parameter ε imposes a limitation "by design" in the obtainable solution similar to the case of mesh size h in the conventional grids. So we can consider ε to represent a grid.

For a given ε, initially very coarse, there is a series of pushes and relabelling steps to convert the initial $1/2\varepsilon$-optimal pseudoflow into a $1/2\varepsilon$-optimal flow. Each step uses as its initial state the final state of the previous step. The result is an "image" of a relaxation mechanism, which is performed by successive local operations.

When the status of $1/2\varepsilon$-optimality is reached, the scaling parameter ε is halved. The state of the problem, defined in this case by the values of the node potentials and reduced costs of the arcs, is transferred to the new grid defined by the halved value of the scaling parameter, now requiring a higher degree of refinement in the process of obtaining the new approximate solution.

Scaling suggests a more abstract concept of a grid, represented in the scaling parameter. With the parallel operations described above, cost scaling could be considered to be an interpretation of a multilevel V-cycle, in which the path from the original to the coarsest grid is done in a straight step, with the return being traced through all the intermediate "grids" defined by the successively halved scaling parameter.

It is conceivable, in order to let the coarse correction scheme come into play, to establish a schematic method similar to the following:

- Relax in the finest grid ($\varepsilon = 1$) a few times. Let the achieved solution be $\mathbf{x}^{(1)}$.
- Double the scaling parameter to move into the next coarser grid 2ε.

- The vector $\mathbf{x}^{(1)}$ now has to be converted into a pseudoflow (restriction). Call the resulting pseudoflow $\mathbf{x}^{(2)}$. The process of obtaining a flow in this level of optimality is equivalent to the solution of the problem at this level. That could be done by initiating a new recursive step, so that the problem at this point is transferred to the 4ε grid
- The solution at the 2ε level can be used as a starting improved point to relax on the finest level.

A relaxation scheme to apply would consist of the push-relabel operation described in Chapter VI. Node potentials and reduced costs are essential elements to update along the grids in the various steps.

G. SUMMARY FOR FURTHER RESEARCH

These points underline the need for investigation of multigrid approaches with other than the conventional grids. The concept of sampling the state of the problem can be extended to "expressing the state of the problem", which does not necessarily involve grid points, but simply levels. Mechanisms like decomposition or scaling are good for handling properties such as arc capacities, to be considered throughout the multigrid development process in optimization. Working on the residual network has not been considered in multigrid. Integrated in a multigrid-type algorithm, we can apply these techniques to perform relaxation at one level of coarsening. The scaling-parameter provides a measure of tracking how fine the relaxation is. Furthermore, it can be used to determine a sufficient degree of relaxation in the level. The idea of pseudoflows that the scaling methods develop is an interesting way to force problems to react in a controlled way to local

in capacity-scaling. All these concepts look promising when integrated in a multigrid scheme.

Also, the work on residual networks provides the checks for optimality (see Chapter II), that have been systematically used in network problems. They involve a follow-up of node potentials and reduced costs. In the transportation problem this is very cheap to compute.

These conclusions should help in opening new ways to approach the multigrid design of algorithms or improve specific areas of those already existing. Multigrid concepts of restriction, interpolation, and the concept of "grid" have been moved to a more abstract area, likely to be productive in the development of new ideas in this field. New optimization tools that engage and fit in the multigrid philosophy have been presented. This opens up a broad field of study and experimentation for future researchers.

APPENDIX A

A. DANTZIG-WOLFE DECOMPOSITION - NUMERICAL EXAMPLE

Consider the following problem

$$\begin{aligned} \min \quad & -2x_1 - x_2 - x_3 + x_4 \\ \text{S.T.} \quad & \\ & x_1 + x_3 \le 2 \\ & x_1 + x_2 + 2x_4 \le 3 \\ & \\ & x_1 \le 2 \\ & x_1 + 2x_2 \le 5 \\ & -x_3 + x_4 \le 2 \\ & 2x_3 + x_4 \le 6 \\ & x_1, x_2, x_3, x_4 \ge 0 \end{aligned}$$

and let the set X consist of the last five constraints. Then

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 2 \end{bmatrix} \qquad b = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

Denote $x^{(j)}$, j = 1,2,..., t the corner points of X, and **s** the slack vector. The problem is reformulated as follows:

$$\min \quad \sum_{j=1}^{t} (cx^{(j)})\lambda_j$$

$$\text{s.t.} \quad \sum_{j=1}^{t} (Ax^{(j)})\lambda_j + s = b$$

$$\sum_{j=1}^{t} \lambda_j = 1 \qquad \lambda_j \geq 0, \qquad s \geq 0$$

To start, we need a first extreme point of X. Choose $\mathbf{x}^{(1)} = (0, 0, 0, 0)$. Let the starting basis consist of **s** and λ_1 . Note that, at this point, the vector of duals, $(\mathbf{w}, \alpha)$ is the zero vector. We will denote $\bar{\mathbf{b}}$ as the updated right hand side in the simplex tableau. Recall that

$$\bar{b} = B^{-1}\begin{bmatrix} b \\ 1 \end{bmatrix}$$

with $\mathbf{B}^{-1}$ being the inverse of the basic submatrix **B** (initially, a 3x3 identity matrix). The initial simplex tableau will adopt the form

	BASIS INVERSE			RHS
z	0	0	0	0
s_1	1	0	0	0
s_2	0	1	0	2
λ_1	0	0	1	1

Also recall that $\mathbf{cx}^{(1)} = 0$.

Subproblem

The subproblem is stated as

$$\begin{aligned} \max \quad & (\mathbf{w}\,\mathbf{A} - \mathbf{c})\mathbf{x} + \alpha \\ \text{s.t.} \quad & \mathbf{x} \in X \end{aligned}$$

which, in terms of the problem data, is expressed

$$\begin{aligned} \max \quad & 2x_1 + x_2 + x_3 - x_4 \\ \text{s.t} \quad & x_1 \leq 2 \\ & x_1 + 2x_2 \leq 5 \\ & -x_3 + x_4 \leq 2 \\ & 2x_3 + x_4 \leq 6 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

The solution to this problem is (2, 3/2, 3, 0), with objective function 17/2, representing the most favourably priced out of the columns corresponding to λ's in the simplex tableau. The lower bound for the objective function is -17/2. (Note: lower bounds are calculated as the current best overall objective function value, minus the solution to the last subproblem, since it gives the maximum of the price-outs).

Master Step

The entering column $C^{(e)}$ is designed to be λ_2, and is obtained in the standard form

$$C^{(e)} = \begin{bmatrix} price\text{-}out \\ y_2 \end{bmatrix}$$

where

$$y_2 = B^{-1}\begin{bmatrix} Ax_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ \frac{7}{2} \\ 1 \end{bmatrix}$$

so

$$C^{(e)} = \begin{bmatrix} \frac{17}{2} \\ 5 \\ \frac{7}{2} \\ 1 \end{bmatrix}$$

After pivoting, we obtain the following tableau

	BASIS INVERSE			RHS
z	-17/10	0	0	-17/5
λ_2	1/5	0	0	2/5
s_2	-7/10	1	0	8/5
λ_1	-1/5	0	1	3/5

So far, the best-known feasible solution to the overall problem is given by

$$x = \lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2 = (4/5, 3/5, 6/5, 0)$$

and the objective function value is -17/5. Also, the vector of duals is (-17/10, 0, 0), as can be seen on the first row of the tableau. Here the first iteration terminates.

Subproblem

Proceeding as in the first iteration, the next subproblem is expressed by

$$\begin{aligned} \max \quad & 3/10\, x_1 + x_2 - 7/10\, x_3 - x_4 \\ \text{s.t.} \quad & \mathbf{x} \in X \quad \text{(for sake of brevity)} \end{aligned}$$

giving $\mathbf{x}^{(3)} = (0, 5/2, 0, 0)$, with objective function value 5/2. λ_3 is introduced. The lower bound now is -17/5 - 5/2 = -59/10 = -5.9.

Master step

The entering column is obtained in similar fashion as the first case

$$y_3 = B^{-1}\begin{bmatrix} Ax_3 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{5} & 0 & 0 \\ -\frac{7}{10} & 1 & 0 \\ -\frac{1}{5} & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ \frac{5}{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{5}{2} \\ 1 \end{bmatrix}$$

The new tableau looks as

	BASIS INVERSE			RHS
z	-6/5	0	-5/2	-49/10
λ_2	1/5	0	0	2/5
s_2	-1/5	1	-5/2	1/10
λ_3	-1/5	0	1	3/5

Now the best-known feasible solution is $x = \lambda_2 x^{(2)} + \lambda_3 x^{(3)} = (4/5, 21/10, 6/5, 0)$, and the objective is -49/10.

Subproblem

$$\begin{aligned} \max \quad & 4/5\, x_1 + x_2 - 1/5\, x_3 - x_4 - 5/2 \\ \text{s.t.} \quad & \mathbf{x} \in X \end{aligned}$$

which solved yields $x^{(4)} = (2, 3/2, 0, 0)$, with objective value 3/5. So λ_4 is introduced. The new lower bound is given by -49/10 - 3/5 = -11/2 = -5.5.

Master step

Proceeding as above, the following tableau is obtained

	BASIS INVERSE			RHS
z	-1	-1	0	-5
λ_2	1/3	-2/3	5/3	1/3
λ_4	-1/3	5/3	-25/6	1/6
λ_3	0	-1	7/2	1/2

the best-known solution is $x = \lambda_2 x^{(2)} + \lambda_3 x^{(3)} + \lambda_4 x^{(4)} = (1, 2, 1, 0)$.

Subproblem

$$\begin{aligned} \max \quad & 0\,x_1 + 0\,x_2 - 0\,x_3 - 3\,x_4 \\ \text{s.t.} \quad & \mathbf{x} \in X \end{aligned}$$

with solution $x^{(5)} = (0, 0, 0, 0)$ and objective 0, which is the termination criterion. Also note that the lower bound is -5 - 0 = -5, equal to the last best objective value (therefore optimal). The optimal solution is then given by $\mathbf{x} = (1, 2, 1, 0)$, with the objective value equal to -5. It is interesting to plot the successive best objective function values (bold letters in the RHS column) and the progress of lower bounds. This is done in Figure 35.

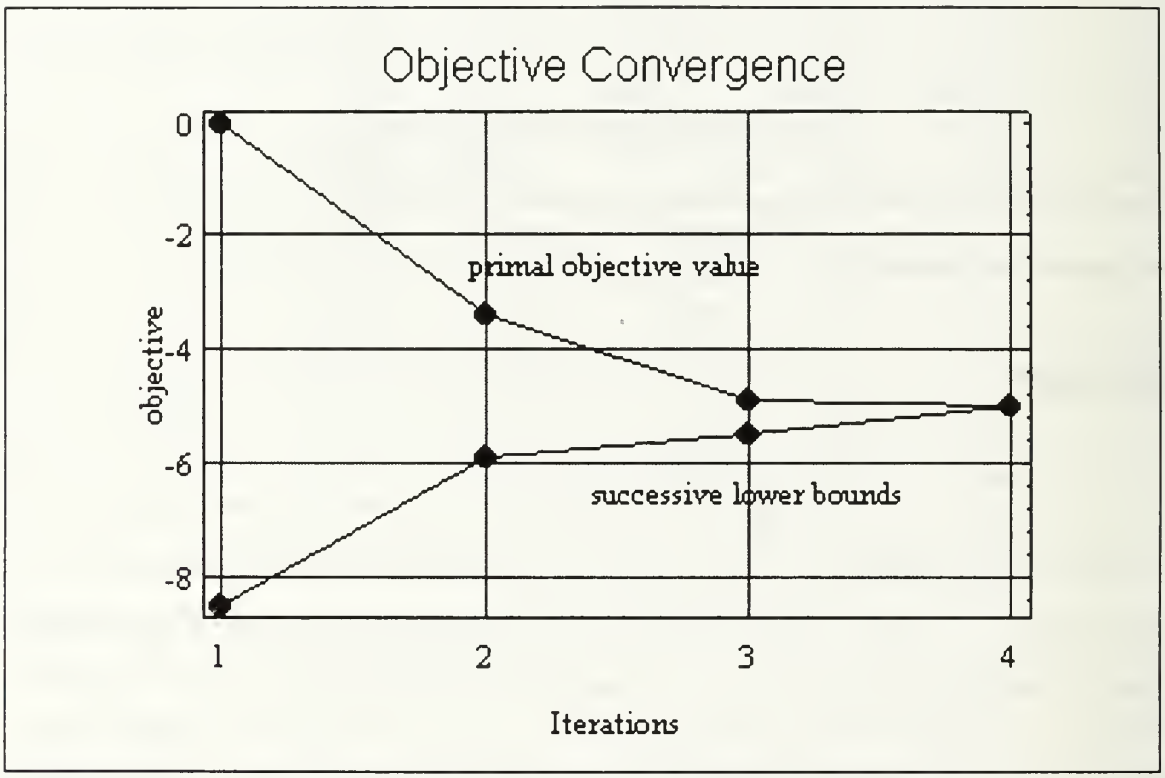

Figure 35.- *Dantzig-Wolfe Decomposition Example. Convergence of Current Objective values and successive Lower Bounds (Minimization).*

APPENDIX B

A. AGGREGATION METHOD - NUMERICAL EXAMPLE

This example is from Balas (1965). The original example has been slightly modified. Figure 36 presents the data of the original problem. The tableau is constructed as described in Chapter III, with costs inside the cells. Rows are associated with supply

	1	2	3	4	5	6	7	8	9	10	11	12	a_i	A_i
1	95	103	65	58	140	133	145	170	180	168	198	188	25	
2	80	88	50	67	125	137	130	155	165	153	183	195	30	**75**
3	**70**	78	**40**	57	**115**	127	**120**	145	155	**143**	**173**	185	20	
4	**30**	38	60	77	55	67	**70**	95	95	**83**	113	111	47	**100**
5	45	53	**45**	62	**40**	52	85	110	110	98	119	**107**	53	
6	**58**	66	**88**	105	80	68	42	67	67	55	85	83	12	
7	70	72	100	117	92	80	**30**	55	65	67	83	95	36	**65**
8	71	79	101	118	67	**55**	55	64	54	**42**	72	**70**	17	
9	137	145	62	**45**	52	40	150	135	125	137	107	95	55	**90**
10	**122**	130	77	60	37	**25**	**135**	120	**110**	122	92	**80**	35	
b_k	22	8	18	42	55	35	50	17	30	18	10	25	330	
B_j	**30**		**60**		**90**		**50**	**65**			**35**			330

Figure 36.- *Balas' Aggregation. Tableau for the Original Problem.*

nodes and columns with destinations.

In the figure, dashed costs lines represent aggregation of nodes. Boldface costs correspond to the cost of each aggregated group, taken to be the minimum of each group.

Supplies and demands are represented in the last two columns and rows, respectively. The supplies (demands) corresponding to the aggregated nodes are placed in the last column (row). They are obtained adding the supply (demand) of the cells forming the aggregated node.

A visualization of the problem is presented in Figure 37. There, dark nodes are destinations. The circles around a group represent the aggregated nodes. The numbers close to each arc are costs. The numbers inside the small circles are the node indices. The figure is illustrative, and, although more than one path could be found between two nodes, each path with different cost, the reader should interpret that the paths giving the costs of the tableau in Figure 36 must be chosen.

Problem II is constructed using the bold-faced data in Figure 36, along with the aggregated demands/supplies. For brevity, the corresponding tableau is not included. Notice that the indices i, j of the aggregated nodes, mentioned in Chapter V, now run along the groups of cells inside each of the dashed rectangles in Figure 36. Solving Problem II the following solution is obtained (expressed in matrix form):

$$\bar{X} = \begin{bmatrix} 15 & 60 & 0 & 0 & 0 & 0 \\ 15 & 0 & 35 & 50 & 0 & 0 \\ 0 & 0 & 0 & 0 & 65 & 0 \\ 0 & 0 & 55 & 0 & 0 & 35 \end{bmatrix}$$

With the resulting solution, Problem III is formed as follows. Pick the groups of cells identified by the indices of those entries in $\bar{\mathbf{X}}$ (i.e. (1,1), (1,2), (2,1), (2,3), (2,4), (3,5), (4,3) and (4,6)). Transfer the whole groups (original problem) into a new tableau, not shown,

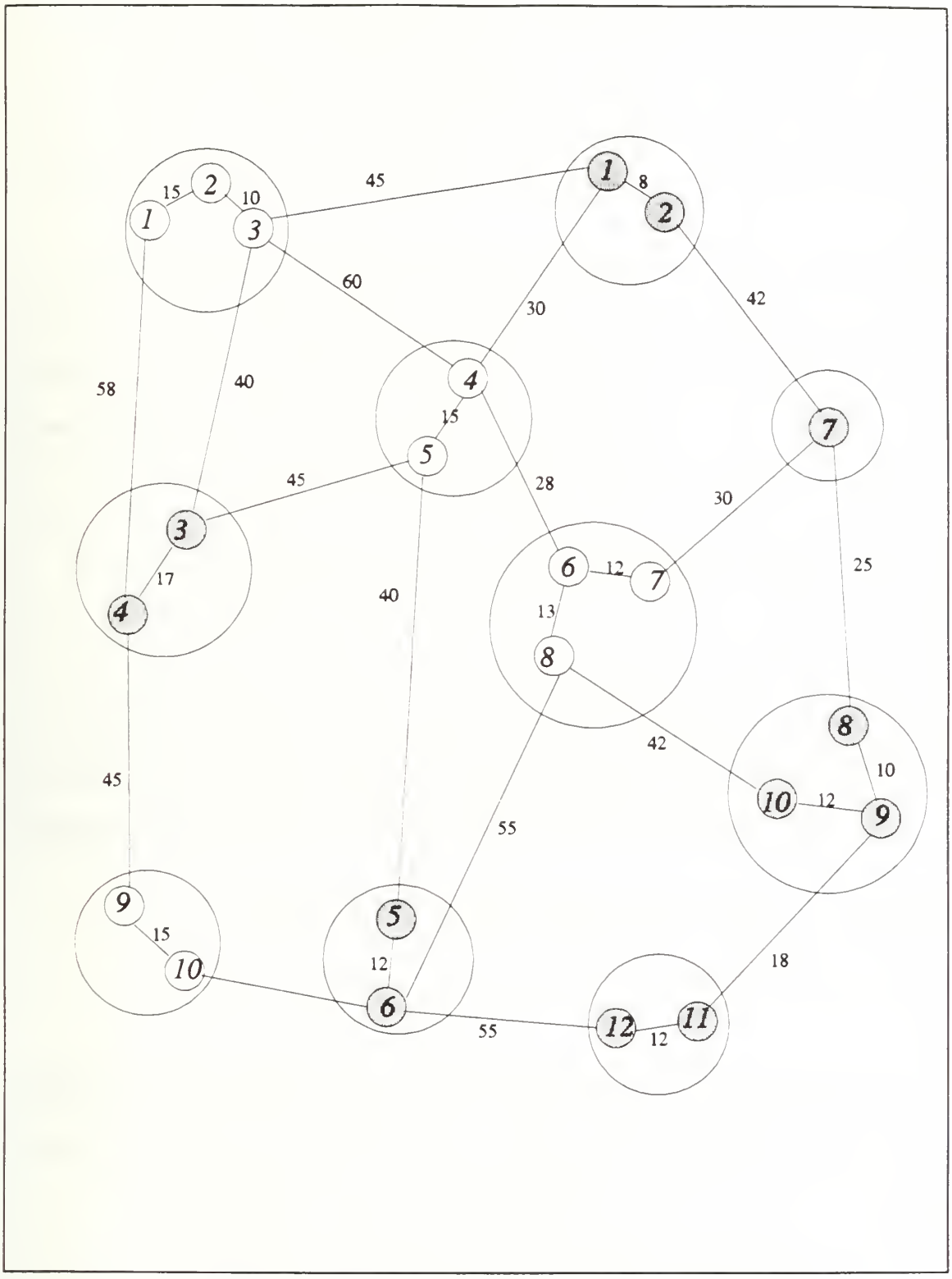

Figure 37.- *"Map" of Origins, Destinations and their aggregation.*

in which the rest of the entries will be given a cost value to force them not to appear in the optimal solution. Solve the problem so constructed. The solution is presented in Figure 38. The blank "blocks" correspond to those entries equal to zero in the solution $\bar{X}$ to Problem II. The last two columns and rows now represent the computed dual values for

	1	2	3	4	5	6	7	8	9	10	11	12	u_h	U_i
1				25									41	
2			13	17									50	**50**
3	7	8	5										40	
4	**15**						**32**						0	**100**
5					**35**			18					15	
6									11	1			-38	
7								17	19				-40	**65**
8										**17**			-51	
9					20	35							27	**90**
10											10	**25**	12	
v_k	30	38	0	17	25	13	70	95	105	93	80	68		
V_l	**38**		**17**		**25**		**70**	**105**			**80**			330

Figure 38.- ***Balas' Aggregation. Solution to Problem III.***

the individual and aggregated nodes, respectively.

Next the D'_{ij} are computed. Recall that those are the values of the slack variables for those blocks not in the optimal solution of Problem II. They are represented in matrix form as

$$D_{ij} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & 40 & 0 & -12 & 43 \\ \mathbf{0} & 13 & \mathbf{0} & \mathbf{0} & -37 & 12 \\ 58 & 109 & 68 & 0 & \mathbf{0} & 28 \\ 57 & 1 & \mathbf{0} & 38 & -22 & \mathbf{0} \end{bmatrix}$$

If all the entries in D'_{ij} were nonnegative, the solution to Problem III would be optimal. Since that is not the case, we need to compute the corresponding values of the slack for all those original cells in the groups affected (i.e. groups (1,5), (2,5), (4,5)). The result of those computations is expressed, in compact form

$$D_{15} = \begin{bmatrix} 34 & 34 & 34 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \end{bmatrix} \quad D_{25} = \begin{bmatrix} 0 & -10 & -10 \\ 0 & -10 & -10 \end{bmatrix} \quad D_{45} = \begin{bmatrix} 13 & -7 & 17 \\ 13 & -7 & 17 \end{bmatrix}$$

Since not all the entries in the above matrices are nonnegative, those blocks of cells must be included in a new iteration to form a new Problem III. The associated tableau is shown in Figure 39. Solving the tableau in Figure 39, the resulting solution is optimal.

	1	2	3	4	5	6	7	8	9	10	11	12	a_i	A_i
1	95	103	65	58									25	
2	80	88	50	67									30	**75**
3	**70**	78	**40**	57									20	
4	**30**	38			55	67	**70**	95	95	**83**			47	**100**
5	45	53			**40**	52	85	110	110	98			53	
6								67	67	55			12	
7								55	65	67			36	**65**
8								64	54	**42**			17	
9					52	40		135	125	137	107	95	55	**90**
10					37	**25**		120	**110**	122	92	**80**	35	
b_j	22	8	18	42	55	35	50	17	30	18	10	25	330	
B_j	**30**		**60**		**90**		**50**	**65**			**35**			330

Figure 39.- ***Tableau for Second Iteration of Problem III.***

APPENDIX C

A. MATHEMATICA FILE MG001.M

```
(* ============================================================
```

This demonstration file shows how the oscillatory components of the error are "quickly" eliminated by a Jacobi iteration technique when solving a suitable system of linear equations.

The system used here is the same system considered in Briggs (1987), i.e., that yielded by the steady-state temperature distribution in a long uniform rod:

$$-u'' + \sigma u = f(x);$$

$$u(0) = u(1) = 0;$$

$$\sigma \geq 0;$$

The domain of the problem is partitioned into 'NN' subintervals. The length of each subintervals is 'h'. The linear system of equations has the form $A x = f$

For generality, the right hand side vector f is a random vector of integer values.

The initial value of the solution is defined as a sum of Fourier "modes", affected by different coefficients. The output is a Table of graphics ("figure"), each representative of the status of the error in the Jacobi process. Applying "ShowAnimation [figure]" to the Table, a movie representation of the error progress is obtained. Limitations in the computer system may force to animate only a certain range of the graphics array.

The residuals 'r = f - A x' are also obtained. A similar graphic for them can be obtained by slight modification of the code.

Reference: "A Multigrid Tutorial", William Briggs, SIAM, 1987.

```
========================================== >> Javier Nieto, 1993 *)
```

```
NN = 48;     NIterations = 200;     h = 1/NN//N;     sigma = 5;   jump = 3;
A = ZeroMatrix[NN+1];
For [i=1, i<= Length[A], ++i, A[[i,i]] = 2 + sigma h^2];
For [i=1, i<= Length[A]-1, ++i, A[[i, i+1]] = A[[i+1, i]] = -1];
A = A/h^2

f = Table[Random[Integer, {-5, 9}], {NN+1}];
c = Table[Random[Real,    {-1, 1}], {NN+1}];

TheRoots = LinearSolve[N[A],N[f]];                          (* True solution *)

Print ["f        = ", ColumnForm[f]];                       Print [" "];
Print ["TheRoots = ", ColumnForm[TheRoots]];   Print [" "];

Diag  = Table [A[[i,i]], {i,1,Length[A]}];          (* Diagonal elements of A *)
DD    = DiagonalMatrix [Diag];                        (* Diagonal Matrix        *)
LL    = Table [If[i > j, -A[[i,j]], 0],
            {i,1,Length[A]},
            {j,1;Length[A]}];                       (* Lower Triangular Matrix *)
UU    = Table [If[i < j, -A[[i,j]], 0],
            {i,1,Length[A]},
            {j,1,Length[A]}];                       (* Upper Triangular Matrix *)
DDinv = DiagonalMatrix [1/Diag];                          (* Inverse of DD *)
PP    = Dot [DDinv, (LL + UU)];                           (* Iteration Matrix *)

                                                  (* Initial value of the solution *)
xInit  = Table[  Sum[k Sin[j k Pi / NN//N], {k,1,10,2}], {j,1,NN+1}];
rInit  = f - Dot[A, xInit];

Print ["Initial x = ", ColumnForm[xInit]];       Print [         " "];
```

```
Print ["Initial r = ", ColumnForm[rInit]];        Print [OutFile, " "];

errors    = {TheRoots - xInit};                    (* Initial value of the error *)
residuals = {rInit};

x         = xInit;      r = rInit;                 (* first values in the loop *)

For [ i = 1, i <= NIterations, ++i,                (* Iterative loop *)
  CompoundExpression [
  x = Dot[Dot[DDinv, (LL + UU)], x]  +  Dot[DDinv, f];
  r = f - Dot[A, x];
  residuals = AppendTo [residuals, r];
  errors    = AppendTo [errors,    TheRoots - x] ];
  Print ["Iteration #: ",i];
];
```

B. MATHEMATICA FILE MG002.M

```
figure = Table [ ListPlot[errors[[i]],
                PlotJoined      -> True,
                PlotRange       -> {-2, 2},
                PlotLabel       -> StringForm ["error ``", i],
                DisplayFunction -> Identity], {i, 1, NIterations, jump}]  ;

fig81first = ListPlot [TheRoots - xInit,
                PlotJoined      -> True,
                PlotRange       -> {-8, 8},
                PlotLabel       -> StringForm ["error 1"],
                DisplayFunction -> Identity];

fig81 = {fig81first, figure[[4]], figure[[8]], figure[[12]], figure[[16]],
        figure[[25]], figure[[35]], figure[[Length[figure]]]};
```

```
figure2 = Table [ ListPlot[residuals[[i]],
                PlotJoined      -> True,
                PlotRange       -> {-400, 400},
                PlotLabel       -> StringForm ["residual “", i],
                DisplayFunction -> Identity], {i, 1, NIterations, jump}]  ;

fig82first = ListPlot [f - Dot[A, xInit],
                PlotJoined      -> True,
                PlotRange       -> {-1500, 1500},
                PlotLabel       -> StringForm ["residual 1"],
                DisplayFunction -> Identity];

fig82 = {fig82first, figure2[[4]], figure2[[8]], figure2[[12]], figure2[[16]],
        figure2[[25]], figure2[[35]], figure2[[Length[figure2]]]};

(* The following lines instruct Mathematica to show a picture using
   several representations of the error term and residuals. The last
   line performs a movie-representation of the error.

   Show [GraphicsArray[Partition[figure81, 2]]]
   Show [GraphicsArray[Partition[figure82, 2]]]
   ShowAnimation [figure[[Range[1,20]]]]        *)
```

C. PROBLEM DATA

```
f = 9  5  -2  -2  4   1   3  -5  -1  -3   3   9  -3  3  4  9  7  9  6  5  3  -4  7  -2  8
    9  5  -2   5  -5   2   4   8   2   9   8   5  -2  5  9 -2
   -5  0   2   0   4   7  -1   8
```

True solution for 'x':

0.02065783515014781
0.03745425067171521
0.05216180823033061

0.06785061971305696
0.0845547320198134
0.0997062287276545
0.1146400740221441
0.1285206208661609
0.1448502141964325
0.1619281803873733
0.1806596365531132
0.1984810658884562
0.2128269767035918
0.2289368349508098
0.2442414345933205
0.2583399609602091
0.2690928709229314
0.2773915553451219
0.2823859679690162
0.28538903069701
0.2868412883700927
0.287613948144673
0.2907468812442201
0.291472581013134
0.2936988723207048
0.2930903087504995
0.28921154185727
0.2837902652892516
0.2798529085677812
0.2743527326374735
0.2716180791304241
0.2686048190242655
0.2644383575705032
0.2573735418579965

0.2499992069220913
0.2392611549873191
0.2255701107673631
0.2101984461280655
0.19615099686665
0.1803590836227403
0.1610523246401646
0.1429631271259921
0.1273543183425618
0.1120218861180346
0.0960645013894233
0.0803155899711188
0.06300486342688363
0.04279267174251583
0.0231073738770337

Initial guess for 'x':

10.35402435050376
18.19077280899915
21.71171356903935
20.33521033480991
14.82383446890182
7.008352827725206
-0.7972048465103275
-6.5
-8.83013611253079
-7.656495855460165
-3.930386807407886
0.7071067811865453
4.511178065381282
6.212778468659391

5.401937904648413
2.59807621135332
-1.000246934522234
-3.985346991432006
-5.258443033643013
-4.413527006719259
-1.857217543664439
1.36939385352089
3.995271049498531
5.
3.995271049498533
1.369393853520896
-1.857217543664434
-4.41352700671926
-5.258443033643003
-3.985346991432011
-1.000246934522246
2.598076211353307
5.401937904648421
6.212778468659399
4.511178065381289
0.7071067811865533
-3.930386807407885
-7.656495855460135
-8.8301361125308
-6.499999999999997
-0.7972048465103328
7.008352827725192
14.82383446890179
20.33521033480992
21.71171356903935

18.19077280899916

10.3540243505038

2.021494950599223*10^-14

-10.35402435050376

LIST OF REFERENCES

Ahuja, Ravindra K., Magnanti, Thomas L., and Orlin, James B., *Network Flows*. Prentice Hall, 1993.

Balas, Egon, "Solution of Large Scale Transportation Problems Through Aggregation", *Operations Research*,13, 1965.

Bazaraa, Mokhtar S., Jarvis, John J. and Sherali, Hanif D., *Linear Programming and Network Flows*, John Wiley and Sons, 1990.

Bertsekas, Dimitri P., *A Distributed Algorithm for the Assigment Problem*. Working Paper, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, 1979.

Bertsekas, Dimitri P., *Linear Network Optimization*, The MIT Press, Cambridge, Massachusetts / London, England, 1991.

Bland, Robert G., and Jensen, David L., "On the Computational Behavior Of a Polynomial-Time Network Flow Algorithm", *Mathematical Programming* **54**, 1-39, 1992.

Bonomi, E. and Lutton, J.L., "The N-city travelling salesman problem: statistical mechanics and the metropolis algorithm", *SIAM Review* **26**, 551, 1984.

Brandt, A., "Levels and Scales", in *Multigrid methods for integral and differential Equations*, *The Institute of Mathematics and its Applications Conference Series (3)*, 1985.

Brandt, A., "Multi-level adaptive solutions to boundary value problems", *Mathematics of Computation*, **31**, 333-390, 1977.

Briggs, William L., *A Multigrid Tutorial*, Society for Industrial and Applied Mathematics, 1987.

Bradley, Gordon H., Brown, Gerald G., and Graves, G.W., "Design and Implementation of Large Scale Primal Transhipment Algorithms", *Management Science*, **24**, 1, 1-31, 1977.

Cavanaugh, Kevin, "Multigrid Approach to the Long Transportation Problem", Master's Thesis, Naval Postgraduate School, September 1992.

Cavanaugh, Kevin, and Henson, Van E., "A Multilevel Cost-Space Approach to solving the balanced Long Transportation Problem", *Proceedings of the 6th Copper Mountain Conference on Multigrid Methods*, NASA Conference Publications **3224**, 61-75, 1992.

Cornett, Annette, "Multigrid Approach to solving the Long Transportation Problem on a Regular Grid in Cost Space", Master's Thesis, Naval Postgraduate School, June 1993.

Edmonds J. and Karp, R.M., "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", *Journal of the ACM* **19**, 248-264, 1972.

Fedorenko, R.P., "The speed of convergence of one iterative process", *USRR Comput. Math. and Math Phys.*, **4**(3), 227-235, 1964.

Gabow, H.N., "Scaling Algorithms for network problems", *Journal of the SIAM*, **9**, 18-27, 1961.

Gabow, H.N., "On the theoretic and practical efficiency of scaling algorithms for network problems", presented at the Fall 1984 ORSA/TIMS meeting (Dallas, TX, 1984).

Geoffrion, A.M. and Graves, G.W., "Multicommodity Distribution System Design By Benders Decomposition, 1973", *Management Science*, **20**, **5**, 822-844, 1974.

Geoffrion, Arthur M., "A Priori Error Bounds for procurement commodity aggregation in Logistics Planning Models", Western Management Science Institute, UCLA, Working Paper No 251.

Gerald, Curtis F. and Wheatley, Patrick O., *Applied Numerical Analysis*, Addison-Wesley Publishing Company, 1989.

Greenberg, Harvey J., "How To Analyze the Results of Linear Programs", *INTERFACES*, **4**, 56-67, 1993.

Hackbusch, Wolfgang, *Multi-Grid Methods and Applications*, Springer-Verlag Berlin, Heidelberg, 1985.

Heiselt, H.L., Pederzoli, G., Sandblom, L., *Continuous Optimization Problems*, Walter de Gruyter, Berlin-New York, 1987.

Kaminsky, R., "Multilevel Solution to the Long Transportation Problem", Master's Thesis, Weizmann Institute of Science, February 1989.

McCormick, S.F., "Multigrid Methods", *Frontiers in Applied Mathematics* **3**, SIAM, Philadelphia, 1987.

Owen, Guillermo, *Game Theory*, Academic Press, 1982.

Pervozvanskii A.A., and Gaitsgori V.G., *Theory of Suboptimal Decisions, Decomposition and Aggregation*, Mathematics and Its Applications (Soviet Series), Kluwer Academic Publishers, 1988.

Roberts, Fred S., *Applied Combinatorics*, Prentice-Hall Inc., 1984.

Röck, H., "Scaling Techniques for Minimal Cost Network Flows", in *Discrete Structures and Algorithms*, edited by V. Page. Carl Hanser, Munich, pp 181-191, 1980.

Ron, Dorit, "Development of Fast Numerical Solvers for Problems in Optimization and Statistical Mechanics", Ph.D. Thesis, Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot 76100, Israel, 1987.

Tardos, É. "A Strongly Polynomial Minimum Cost Circulation Algorithm", *Combinatorica* **5**, 247-255, 1985.

Wesseling, Pieter, *An Introduction to Multigrid Methods*, John Wiley & Sons, 1992.

Wolfram, Steven, *Mathematica, A System for Doing Mathematics by Computer, Windows version 2.2.*

Zipkin, Paùl H., "Aggregation in Linear Programming", Ph.D. Dissertation, Yale University, 1977.

INITIAL DISTRIBUTION LIST

1	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3	Professor Richard Franke Department of Mathematics Naval Postgraduate School Monterey, CA 93943-5000	1
4	Professor Van Emden Henson, Code MA/Hv Department of Mathematics Naval Postgraduate School Monterey, CA 93943-5000	2
5	Professor Gordon H. Bradley, Code OR/Bz Department of Operations Research Naval Postgraduate School Monterey, CA 93943-5000	3
6	Professor Craig W. Rasmussen, Code MA/Ra Department of Mathematics Naval Postgraduate School Monterey, CA 93943-5000	2

7	LCdr Kevin J. Cavanaugh, USCG 1 Norman Drive Gales Ferry, CT 06335	1
8	Lt. Annette P. Cornett 1606 Kickapoo Road Pueblo, CO 81001	1
9	LCdr Javier Nieto G.I.M.O. Arturo Soria, 287 28033 Madrid, Spain	4